# Asterisk™: The Future of Telephony

Second Edition

*Jim Van Meggelen,*
*Leif Madsen, and Jared Smith*

O'REILLY®

# Asterisk™: будущее телефонии

Второе издание

*Джим Ван Меггелен,*
*Лейф Мадсен и Джаред Смит*

*Санкт-Петербург — Москва*
*2009*

Джим Ван Меггелен, Лейф Мадсен, Джаред Смит

# Asterisk™: будущее телефонии, 2-е издание

Перевод Н. Шатохиной


| | |
|---|---|
| *Главный редактор* | *А. Галунов* |
| *Зав. редакцией* | *Н. Макарова* |
| *Выпускающий редактор* | *Л. Пискунова* |
| *Научный редактор* | *А. Бирюков* |
| *Редактор* | *Е. Тульсанова* |
| *Корректор* | *Е. Бекназарова* |
| *Верстка* | *Н. Пискунова* |

*Меггелен Дж., Мадсен Л., Смит Дж.*
Asterisk™: будущее телефонии, 2-е издание. – Пер. с англ. – СПб: Символ-Плюс, 2009. – 656 с., ил.

ISBN13: 978-5-93286-128-8
ISBN10: 5-93286-128-2



Asterisk – система телефонии, обладающая такими фантастическими возможностями, о которых обычная офисная коммутируемая АТС может только мечтать. Голосовая почта, конференц-связь, очереди вызовов и агенты, музыка во время ожидания и парковка вызовов – это лишь часть функций, обеспечиваемых Asterisk. Описать в одной книге всю функциональность этой необыкновенно гибкой системы невозможно, но вы получите исчерпывающее представление о базовых функциях Asterisk. А проявив свои творческие способности, вы сможете создать офисную АТС, настроенную целиком под ваши нужды с учетом абсолютно всех требований пользователей.



**ISBN13: 978-5-93286-128-8**
**ISBN10: 5-93286-128-2**
**ISBN 0-596-51048-0 (англ)**





Издательство «Символ-Плюс». 199034, Санкт-Петербург, 16 линия, 7,
тел. (812) 3245353, www.symbol.ru. Лицензия ЛП N 000054 от 25.12.98.

Налоговая льгота – общероссийский классификатор продукции
ОК 005 93, том 2; 953000 - книги и брошюры.

Подписано в печать 29.09.2008. Формат 70x100 $^{1}/_{16}$. Печать офсетная.
Объем 41 печ. л. Тираж 2000 экз. Заказ N
Отпечатано с готовых диапозитивов в ГУП «Типография «Наука»
199034, Санкт-Петербург, 9 линия, 12.

*Эта книга посвящается Ричу Адамсону*
*(1947–2006).*
*Спасибо вам за то, что открыли для нас*
*значимость сообщества.*

# Оглавление

# Предисловие

Когда-то давным-давно жил-был мальчик

…и был у него компьютер

…и телефон.

Так все просто начиналось, а к скольким неприятностям это привело!

Не так давно дистанционная передача данных и голоса, а также программное обеспечение были коммерческими продуктами и сервисами. Некоторые из них контролировались клубом избранных компаний, создающих технологии, другие – компаниями, использующими продукты для предоставления сервисов. К концу 1990-х годов распространение Интернета сделало дистанционную передачу данных общедоступной. Цены упали. Возникли новые прогрессивные технологии, сервисы и компании. В то же время появилась полностью открытая программная платформа Linux (или GNU/Linux), ставшая результатом трудов пионеров-создателей свободного программного обеспечения Ричарда Столлмана (Richard Stallman), Линуса Торвалдса (Linus Torvalds) и многих других. Тем не менее голосовая связь хоть и была повсеместно распространенной, но по-прежнему оставалась коммерческой. Почему? Может быть, потому, что голосу в старой телефонной сети общего пользования не хватало очарования и блеска многообещающей новой глобальной сети связи. Или, возможно, потому, что телефон просто не мог так же эффективно обеспечить развлечение для взрослых. Какой бы ни была причина, одно оставалось очевидным: телефонная связь с открытым исходным кодом была распространена практически так же широко, как и защищенное от несанкционированного копирования ПО с открытым исходным кодом.

Востребованность (а в некоторых случаях просто дешевизна) – действительно двигатель прогресса. В 1999 году, когда я основал компанию Linux Support Services (Служба поддержи Linux) с целью предложить бесплатную и коммерческую техническую поддержку для Linux, мне понадобилась (или, по крайней мере, мне так казалось) телефонная система для обеспечения круглосуточного обслуживания клиентов. Идея заключалась в том, что люди смогут звонить, вводить идентификационную информацию пользователя и оставлять сообщение. В свою очередь, система отправляла бы сообщение на пейджер техническому специалисту, что обеспечивало бы ответ на запрос клиента в максимально короткие сроки. Поскольку стартовый капитал моей компании был около $4000, я не мог позволить себе приобрести телефонную сис-

тему, которая могла бы обеспечить реализацию задуманного сценария. Будучи пользователем Linux с 1994 года и уже занимаясь разработкой ПО с открытым исходным кодом, поучаствовав в проектах l2tpd, Gaim и cheops и не имея никого, кто мог бы объяснить сложность подобной задачи, я решил, что просто создам собственную телефонную систему, позаимствовав оборудование в компании Adtran, где проходил стажировку. Я фантазировал, что, сумев передать телефонный вызов в компьютер, смогу делать *все что угодно*. Кстати, из этого предположения был выведен официальный девиз Asterisk (у любого достаточно большого и значимого проекта должен быть девиз):

*Это только программа!*

Хорошо это или плохо, но я всегда мыслил глобально. С самого начала я задумывал Asterisk как систему, которая сможет делать *все* в телефонии. Имя Asterisk было выбрано потому, что это одновременно и кнопка *, которая есть на любом обычном телефоне, и символ подстановки «все» в Linux (например, `rm -rf*`).

Итак, в 1999 году я создал бесплатную платформу для телефонной связи, которую выложил в Сети, и продолжил зарабатывать на жизнь, оказывая техническую поддержку Linux. Однако к 2001 году по мере спада экономики стало очевидным, что компания Linux Support Services была бы более прибыльной, если бы мы занимались исключительно Asterisk, а не общей технической поддержкой Linux. В тот год мы познакомились с Джимом Диксоном (Jim Dixon) по прозвищу Dude, создателем проекта телефонной связи Zapata. Его восхитительный труд был фантастическим дополнением к Asterisk и предоставил нам бизнес-модель для начала более целенаправленной работы над Asterisk. После создания вместе с Dude нашей первой интерфейсной PCI-платы для телефонной связи стало очевидным, что Linux Support Services не лучшее название для телефонной компании. Мы решили назваться Digium (но это совсем другая история, и об этом трудно рассказать). С началом распространения технологии VoIP (Voice over IP – передача голоса по IP-протоколу), с революционным переходом от старых коммутируемых сетей к новым сетям на базе IP-протокола все изменилось.

Итак, как мы уже говорили, большинству людей телефоны явно не кажутся чем-то впечатляющим. Конечно, очень немногие могли разделить мой восторг от гудка, зазвучавшего в телефоне, подключенном к ПК. Однако те, кто *действительно* проникся этим чувством, *на самом деле* «заболели» телефонами. И с помощью Интернета эти несколько человек смогли теперь объединить свои усилия и воплотить свою одержимость в обычный практический проект на пользу людям.

Сказать, что индустрия связи созрела для решения с открытым исходным кодом, все равно что ничего не сказать. Телекоммуникации имеют громадный рынок из-за повсеместной востребованности телефонов на работе и в личной жизни. Непосредственный рынок сбыта телеком-

муникационных продуктов имеет аудиторию, обладающую знаниями в сфере высоких технологий и желающую и способную внести свой вклад в ее развитие. Люди требуют безгранично настраиваемых решений. Подобные решения для коммерческой телефонной связи очень дорогие. Создание Asterisk было просто искрой в этом баке, наполненном бензином.

Asterisk находится на пересечении эпох (коммерческий → с открытым исходным кодом; коммутируемый → VoIP; только голос → голос, видео и данные; цифровая обработка сигналов → обработка мультимедийных данных на стандартных массовых серверах; централизованный каталог → одноранговая связь), упрощая переход между ними за счет предоставления обратной совместимости с использовавшимися ранее технологиями. Asterisk поддерживает любые устройства, начиная от телефонов с импульсным набором 1960-х годов и заканчивая самыми современными беспроводными VoIP-устройствами, и обеспечивает любую функциональность, от простой последовательной коммутации до поддержки технологий Bluetooth и DUNDi. Но самое главное, Asterisk является свидетельством того, что сообщество заинтересованных людей и компаний может, объединив усилия, создать проект с поистине безграничными возможностями, что не под силу осуществить одному человеку или компании. Особую благодарность за то, что Asterisk стала реальностью, я хотел бы выразить Линусу Торвальдсу, Ричарду Столлману, всему сообществу разработчиков Asterisk и тому человеку, который придумал Red Bull.

Так каким будет дальнейший путь развития Asterisk? Вспомним историю ПК. Впервые появившись в 1980-х годах, он обладал довольно ограниченными возможностями. Наверное, на нем можно было создать таблицу, набрать текст, в общем, немногое. Со временем, однако, его открытая архитектура привела к снижению цен и появлению новых продуктов, позволяющих постепенно распространять сферу его применения. ПК начал неуклонно вытеснять миникомпьютер, а затем и ЭВМ. Теперь даже суперкомпьютеры компании Cray создаются с архитектурой x86 на базе Linux. Я уверен, что Asterisk ожидает то же самое. Сегодня Asterisk уже обслуживает бóльшую часть телефонии. Кто знает, каких масштабов это достигнет завтра?

Итак, чего вы ждете? Читайте, изучайте и участвуйте в создании будущего открытых телекоммуникаций, присоединившись к революционному движению Asterisk!

*– Марк Спенсер*

# Введение

Эта книга для всех, кто начинает свое знакомство с Asterisk™.

Asterisk – это конвергированная платформа для телефонии с открытым исходным кодом, разработанная, главным образом, для выполнения на Linux. Более чем 100-летний опыт телефонной связи позволил создать надежный пакет тесно интегрированных телекоммуникационных приложений. Мощь Asterisk – в ее настраиваемой природе в сочетании с не имеющим аналогов соответствием стандартам. Ни одна другая офисная АТС не предоставляет такие широкие возможности по вариантам ее развертывания.

Такие приложения, как голосовая почта, конференц-связь, очереди вызовов и агенты, музыка во время ожидания и парковка вызовов, – все это стандартные функции, встроенные непосредственно в программное обеспечение. Более того, Asterisk может интегрироваться с другими бизнес-технологиями такими способами, о которых закрытые узкоспециализированные офисные АТС могут только мечтать.

Новому пользователю Asterisk может показаться достаточно устрашающей и сложной, поэтому документация так важна для ее роста. Документация облегчает освоение системы и помогает людям увидеть ее возможности.

Выпущенная при поддержке O'Reilly Media, книга «Asterisk: будущее телефонии» была создана под впечатлением от работы, начатой Asterisk Documentation Project. Мы прошли длинный путь, и эта книга является реализацией желания создать документацию, которая представляла бы самые фундаментальные элементы Asterisk: то, что необходимо знать каждому, кто начинает работать с Asterisk. Эта книга – первый том будущей, мы уверены в этом, огромной библиотеки знаний по Asterisk.

Эта книга написана для и при участии сообщества разработчиков Asterisk.

## Целевая аудитория

Данная книга рассчитана на новичков в Asterisk, но мы предполагаем, что вы хорошо знакомы с основами администрирования Linux, построением сетей и другими ИТ-дисциплинами. Если нет, рекомендуем изучить богатую и замечательную библиотеку книг O'Reilly, посвященных этим вопросам. Также предполагается, что вы практически не знакомы с телекоммуникациями: как с традиционной коммутируемой телефонной связью, так и с новым миром передачи голоса по IP-протоколу.

## Структура книги

Данная книга организована следующим образом:

*Глава 1. Революция в телефонии*

Здесь мы начинаем свой путь к знаниям. Asterisk собирается изменить мир телефонной связи, и в данной главе мы обсуждаем основания нашей веры в это.

*Глава 2. Подготовка системы к установке Asterisk*

Рассматриваются некоторые технические вопросы, о которых следует помнить при проектировании системы телефонной связи. Бóльшую часть изложенного материала можно пропустить, если не терпится перейти сразу к установке, но данные концепции важно понимать, если вы планируете когда-нибудь вводить систему Asterisk в производственную эксплуатацию.

*Глава 3. Установка Asterisk*

Рассматривает, как приобрести, скомпилировать и установить Asterisk.

*Глава 4. Исходная конфигурация Asterisk*

Описывается исходная конфигурация Asterisk. Здесь будут рассмотрены основные конфигурационные файлы, которые должны существовать для описания каналов и функций, доступных вашей системе.

*Глава 5. Основы диалплана*

Представляет сердце Asterisk – диалплан.

*Глава 6. Дополнительные концепции диалплана*

Рассматриваются некоторые более сложные концепции диалплана.

*Глава 7. Что такое телефония*

Отступая от Asterisk, в данной главе мы обсуждаем некоторые наиболее важные технологии, используемые в телефонной сети общего пользования.

*Глава 8. Протоколы для VoIP*

Продолжая обсуждение традиционной телефонии, здесь мы рассматриваем технологию передачи голоса по протоколу IP.

*Глава 9. Шлюзовой интерфейс Asterisk (AGI)*

Представляет один из самых удивительных компонентов – шлюзовой интерфейс Asterisk. Используя языки программирования Perl, PHP и Python, мы демонстрируем, как можно с помощью внешних программ вводить практически безграничные функциональные возможности в свою офисную АТС.

*Глава 10. Интерфейс Asterisk Manager (AMI) и Adhearsion*

Описывается возможность подключения внешних приложений к Asterisk для управления или отслеживания различных аспектов

системы. Также в данную главу включено краткое введение в инфраструктуру Adhearsion.

*Глава 11. Инфраструктура Asterisk GUI*

Инфраструктура Asterisk GUI, появившаяся в Asterisk 1.4, – это среда, с помощью которой веб-разработчики получили возможность создавать графические интерфейсы с минимальным вмешательством в стандартные конфигурационные файлы.

*Глава 12. Интеграция с реляционными базами данных*

Поэтапно рассматривается настройка Asterisk для работы с базами данных по стандарту ODBC.

*Глава 13. Управление системой Asterisk*

Обсуждаются вопросы, касающиеся предпочтительных способов управления телефонной системой Asterisk, включая записи CDR, журналы регистрации и приглашения.

*Глава 14. Попурри*

Кратко рассматривается то, что, по сути, можно назвать рогом изобилия богатых и невероятных возможностей и функций, составляющих феномен Asterisk.

*Глава 15. Asterisk – будущее телефонии*

Предсказывание будущего, в котором телефония с открытым исходным кодом полностью преобразит отрасль, отчаянно нуждающуюся в революционных изменениях.

*Приложение A. Каналы VoIP*

*Приложение B. Справочник по приложениям*

*Приложение C. Справочник по AGI*

*Приложение D. Конфигурационные файлы*

*Приложение E. Функции диалплана Asterisk*

*Приложение F. Команды интерфейса Asterisk Manager*

*Приложение G. Пример func_odbc*

## Программное обеспечение

Основное внимание данная книга уделяет документированию Asterisk версии 1.4; однако многие соглашения и информация в данной книге являются универсальными и не относятся к какой-либо конкретной версии. Для выполнения и тестирования Asterisk мы использовали операционную систему Linux, тяготея к синтаксису Red Hat. Мы решили, что, хотя основанные на Red Hat дистрибутивы, возможно, не являются самыми популярными, тем не менее их компоновка и утилиты хорошо знакомы многим опытным администраторам Linux.

## Принятые соглашения

В данной книге действуют следующие соглашения о шрифтовом оформлении:

*Курсив*

Применяется для выделения новых терминов и URL.

`Моноширинный шрифт`

Предназначен для команд, опций, параметров и аргументов, подставляемых в команды.

**`Моноширинный полужирный`**

Обозначает команды или другой текст, вводимый пользователем буквально. Также используется для выделения фрагментов в листингах кода.

*`Моноширинный курсив`*

Выделяет текст, который должен быть заменен определяемыми пользователем значениями.

`[ Ключевые слова и прочее ]`

Показывает необязательные ключевые слова и аргументы.

`{ выбор-1 | выбор-2 }`

Обозначает *выбор-1* или *выбор-2*.

Так отмечаются подсказки, советы или примечания.

Это предупреждения или предостережения.

## Использование примеров кода

Данная книга призвана помочь вам в вашей работе. Вообще говоря, код из нее вы можете свободно использовать в своих программах и документации. Не надо обращаться к нам за разрешением на использование небольших частей кода, например, при написании программы, в которой применяется несколько блоков кода из этой книги. А вот продажа или распространение CD-ROM с примерами из книг O'Reilly требует специального разрешения. Можно свободно ссылаться на книгу и цитировать примеры кода, но для включения больших частей кода из нее в документацию вашего продукта требуется наше согласие.

Будем признательны, но не настаиваем на указании авторства. Обычно ссылка на источник включает название, автора, издателя и ISBN. На-

пример: «*Asterisk: The Future of Telephony,* Second Edition by Jim Van Meggelen, Leif Madsen, and Jared Smith. Copyright 2007 O'Reilly Media, Inc., 978-0-596-51048-0».

Если вам кажется, что использование вами примеров кода выходит за рамки законного использования или разрешений, оговоренных выше, не стесняйтесь, обращайтесь к нам по адресу permission@oreilly.com.

## Safari® Books Online


Если на обложке книги есть пиктограмма Safari® Books Online, это означает, что книга доступна в Сети посредством O'Reilly Network Safari Bookshelf.

Safari предлагает намного лучшее решение, чем электронные книги. Это виртуальная библиотека, позволяющая без труда находить тысячи превосходных технических книг, копировать и использовать примеры кода, скачивать главы и быстро находить ответы, когда требуется самая точная и свежая информация. Она свободно доступна по адресу *http://safari.oreilly.com*.

## Контактная информация

Пожалуйста, направляйте комментарии и вопросы, касающиеся данной книги, издателю:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
(800) 998-9938 (в США или Канаде)
(707) 829-0515 (международный или местный)
(707) 829-0104 (fax)

Для данной книги создана веб-страница, на которой представлен список опечаток, примеры и другая дополнительная информация. Ее можно найти по адресу

*http://www.oreilly.com/catalog/9780596510480*

Вопросы и комментарии по данной книге присылайте по электронной почте:

bookquestions@oreilly.com

Для получения более подробной информации о книгах, конференциях, ресурсах и сети O'Reilly Network посетите веб-сайт издательства:

*http://www.oreilly.com*

# Благодарности

Прежде всего мы должны поблагодарить нашего фантастического редактора Майкла Лукидеса (Michael Loukides), который давал бесценные отзывы и находил чрезвычайно тактичные способы попросить нас переписать раздел (или главу), когда это было необходимо, и подать это как нашу собственную идею. Майк поддерживал нас, когда у нас опускались руки, и возвращал на землю, когда мы вели себя слишком самонадеянно. Вы мастер, Майк, и видя, как много книг получило ваше редакторское благословение, мы понимаем, почему O'Reilly Media достигло такого успеха.

Также спасибо Сандерсу Клейнфельду (Sanders Kleinfeld), нашему литературному редактору, Лорел Рума (Laurel Ruma), нашему выпускающему редактору, и всем остальным неназванным героям производственного отдела O'Reilly. Эти ребята взяли нашу книгу и превратили ее в *книгу O'Reilly*.

Все сообщество разработчиков Asterisk должно быть благодарно Джиму Диксону (Jim Dixon) за создание первых аппаратных интерфейсов телефонии с открытым исходным кодом, положивших начало революции, и за то, что он полностью передал свое творение сообществу.

Спасибо Тиму О'Рейлли (Tim O'Reilly) за предоставленный шанс написать эту книгу.

Большое спасибо нашим самым благородным и беспощадным рецензентам:

- Ричу Адамсону (Rich Adamson), президенту Network Partners Inc., за энциклопедические знания технологии PSTN и неустанное желание делиться опытом. Ваше великодушие даже в условиях, когда судьба послала вам ужасное испытание, вдохновляет всех нас[1].
- Тилгману Лешеру (Tilghman Lesher) за тщательнейшее чтение нашей книги и особое внимание к приложениям B и F, а также за некоторые замечательные новые приложения и функции Asterisk.
- Эндрю Колсмиту (Andrew Kohlsmith) за помощь в написании раздела главы 14, посвященного хранению голосовой почты по протоколу IMAP.
- Дэвиду Трою (David Troy) за техническую рецензию, за AstManProxy и за перенос Asterisk на Roomba (первая офисная АТС, выполняемая в пылесосе!).

[1] В декабре 2006 года Рич скончался, его двухлетняя борьба с раком пришла к неутешительному концу. Рич присылал свои сообщения в рассылку Asterisk Users до ноября того года. Он отдавал силы сообществу до самого конца, поэтому мы посвятили эту книгу ему.

- Мэттью Гасту (Matthew Gast), соавтору O'Рейлли, за то, что прочитал нашу книгу от корки до корки и дал нам всеобъемлющую рецензию, а также за работу «T1, The Definitive Guide».
- Доктору Эдварду Гаю III (Edward Guy III) за исчерпывающие и острые, как бритва, оценки всех до одной глав первого издания и за борьбу за Asterisk.
- Кристиану Килхофнеру (Kristian Kielhofner), президенту Kris-Companies и создателю AstLinux, за самый лучший дистрибутив AstLinux.
- Расселлу Брайанту (Russell Bryant) за быстрые и полезные ответы на наши вопросы.
- Джошуа Колпу (Joshua Colp) за помощь в настройке производительности и ответы на многочисленные вопросы.
- Кевину Флемингу (Kevin Fleming), который уважаем (осмелимся сказать, любим) всеми за поднятие планки и за мастерство.
- Брайану Капучу (Brian Capouch) – он рассказывает о том, что возможно, а потом идет и делает это.
- Стивену Улеру (Stephen Uhler) за успешный перенос Zaptel на Solaris и за предоставление нам ряда бесценных примеров.
- Джейсону Паркеру (Jason Parker) за то, что он не «чайник».
- Экке Лу (Ekke Loo) за разгром главы, посвященной базам данных.
- Яну Дарвину (Ian Darwin) за то, что помог выразить некоторые наши мысли более лаконично, и за вишнево-красный дисковый телефон (который работает с Asterisk!).
- Джоэлу Сиско (Joel Sisko), генеральному директору iConverged, за ваши всесторонние знания телефонии и электрических монтажных схем.

Последнее и самое важное спасибо Марку Спенсеру (Mark Spencer) за Gaim (недавно переименованный в Pidgin, *www.pidgin.im*), Asterisk и DUNDi и за предоставление своих творений сообществу разработчиков продуктов с открытым исходным кодом.

## Джим Ван Меггелен

Для меня все началось весной 2004 года, когда я сидел за своим столом в отделе технической поддержки телефонной компании, в которой проработал около 15 лет. Не имея возможности применять приобретенные навыки, я проводил время в попытках понять, на что будет похожа оставшаяся часть моей профессиональной деятельности. Телекоммуникационная отрасль перестала быть любимицей инвесторов и превратилась в посмешище, о котором не знал только ленивый. Я должен быть счастлив, что оставался среди тех немногих, у кого еще была работа, но какой неблагодарной, бесцельной была эта работа! Мы понимали, почему наша отрасль разрушилась: продаваемые нами про-

дукты не могли обеспечить решения, необходимые нашим клиентам, даже несмотря на обещания обратного. Им не хватало гибкости, и их стоимость абсолютно не соответствовала той функциональности, которую они предлагали (или, точнее, не предлагали). И не было никаких признаков того, что в ближайшем будущем что-либо изменится.

Долгие годы я мечтал об офисной АТС с открытым исходным кодом, но на самом деле не представлял, как такое возможно, и отказался от этой идеи несколькими годами раньше. Я знал, что для достижения успеха офисная АТС с открытым исходным кодом должна была бы эффективно объединить миры традиционной и сетевой телефонии. Но мне никогда не удавалось найти что-либо, пригодное для реализации этой задачи.

Как-то раз в один замечательный весенний день я без особого энтузиазма выполнил поиск в Google по фразе «телефония с открытым исходным кодом» и обнаружил блестящее будущее телефонии: Asterisk, офисную АТС с открытым исходным кодом на базе Linux[1].

Это было оно – то самое, о чем я мечтал так много лет. Я понятия не имел, каким образом собираюсь принять участие в этом, но знал: телефония с открытым исходным кодом приведет к необходимой и благотворной революции в телекоммуникационной отрасли и так или иначе я буду ее частью.

Для меня, скорее системного интегратора, чем разработчика, необходимо было найти способ войти в сообщество. Недостатка в разработчиках не было, но был очевидный недостаток в документации. Здесь, казалось, я мог помочь. Я знал, как писать, я знал офисные АТС, и я отчаянно хотел поговорить об этом чуде, которое вдруг сделало телефонию снова интересной.

Даже если мой вклад в эту книгу совсем невелик, надеюсь, вы ощутите мое восторженное отношение к телефонии с открытым исходным кодом. Это удивительный дар, но также и невероятная ответственность. Потрясающий вызов. Грандиозный шанс. Бесподобное удовольствие!

Прежде всего я должен поблагодарить Лейфа и Джареда за приглашение участвовать в Asterisk Documentation Project. Работать с вами было огромным удовольствием, и я не устаю удивляться, как замечательно наши личностные качества и навыки дополняют друг друга. Поистине гармоничная группа, не так ли? Также спасибо Фигменту (Figment) за набор текста.

[1] Чтобы почувствовать, насколько велик феномен Asterisk, наберите в Google «офисная АТС». Увидев результаты, имейте в виду, что традиционные офисные АТС оцениваются миллиардами долларов. Крупными игроками являются компании Avaya, Nortel, Siemens, Mitel, Cisco, NEC и многие-многие другие. Что-то подсказывает, что их мало волнует их ранг в поисковой системе Google. Но мы уверены, что как культурный барометр это имеет значение.

Моей жене Килли (Killi) и детям Кааре (Kaara), Джунасу (Joonas) и Джузепу (Joosep) (которые не забывали навещать меня, если я пропадал в своей подземной берлоге слишком надолго): вы источник вдохновения для меня. Ваша любовь – это то, что не дает моему огню погаснуть, и я благодарен вам.

Несомненно, я должен поблагодарить моих родителей, Джека (Jack) и Мартини (Martiny), за их неизменную веру в меня независимо от того, как много правил я нарушил. Через несколько лет мои дети подрастут – тогда придет ваш черед смеяться!

Марку Спенсеру: спасибо за все, за что вам благодарны все остальные, но также лично от меня спасибо за то время, которое вы великодушно отдаете сообществу разработчиков Asterisk. Группа пользователей Asterisk Toronto (*http://www.taug.ca*) сделала качественный скачок вперед после вашего выступления, и это событие навсегда останется частью нашей истории. О да, и спасибо за пиво. :-)

Наконец, спасибо сообществу разработчиков Asterisk. Эта книга – наш подарок вам. Надеемся, вы будете читать ее с таким же удовольствием, с каким мы писали ее.

## Лейф Мадсен

Дорога к этой книге была долгой, на ее создание ушло около трех лет. Когда я начал использовать Asterisk, вероятно, как и вы, я ничего не знал о ней, знал очень немного о традиционной телефонии и даже еще меньше о Voice over IP. Я с головой окунулся в этот новый и волнующий мир и вбирал все, что мог. В течение двух месяцев практики, когда я не мог быстро определиться, чем буду заниматься, я впитывал максимум знаний, задавая вопросы, пробуя и открывая возможности системы. К сожалению, не было практически никакой документации по Asterisk. Мне удалось лишь отыскать примеры диалплана Джона Тодда (John Todd) и получить ответы на вопросы от Брайана К. Веста (Brian K. West) в IRC. Конечно, так не могло долго продолжаться.

Не будучи хорошим кодировщиком, я искал другие способы пригодиться сообществу. А что кодировщики не любят больше всего? Документацию! Так я начал работу над The Asterisk Documentation Assignment (TADA), базовым конспектом с небольшим объемом информации для начала книги.

Вскоре после опубликования этого труда на своем веб-сайте я получил письмо от очень смышленого парня по имени Джаред Смит. Он тоже жаждал создать для сообщества «бумажную» книгу, и мы с замиранием сердца запустили проект Asterisk Documentation Project. Джаред создал простой веб-сайт по адресу *http://www.asteriskdocs.org*, CVS-сервер (Concurrent Versions System – система контроля версий) и выложил самую первую версию книги в формате DocBook для Asterisk. С этого момента мы начали сбор информации, и вскоре в этот процесс включились члены сообщества.

В июне 2004 года в рассылках стал появляться энергичный малый по имени Джим Ван Меггелен. Он присылал массу информации и документации. Несомненно, именно такого парня не хватало нашей команде! Джим обладал видением и энергией, которые расшевелили нас с Джаредом и заставили взяться за что-то более великое. Джим принес с собой годы опыта и талант писателя, о которых мы не могли даже мечтать.

Сформировав ядро команды по созданию документации, мы приступили к выполнению плана по написанию книг по Asterisk, которые в конечном итоге образуют полную библиотеку и богатый источник информации. Фактически данная книга – начало реализации этой мечты.

Прежде всего я должен поблагодарить своих родителей, Рика (Rick) и Кэрол (Carol), за то, что они всегда поддерживают мои начинания, позволяя реализовывать мои мечты, и всегда ставят мои нужды превыше собственных. Без их видения, понимания и проницательности было бы невозможно достичь того, чего я достиг. Я очень люблю вас обоих!

Хотел бы выразить благодарность Феликсу Карапаике (Felix Carapaica) и Биллу Фаркасу (Bill Farkas) из Шериданского института технологий за их преданность продвижению знаний. Они дополнили мои предыдущие знания и чрезвычайно расширили мои представления о маршрутизации и о телекоммуникациях.

Хотелось бы отметить очень многих, но особенно важную роль сыграли и продолжают играть, оказав наибольшее влияние на формирование моего понимания Asterisk, Джошуа Колп, Тилгман Лешер, Рассел Брайант, Стив Мерфи (Steve Murphy), Олли Йоханссон (Olle Johansson), Стивен Сокол (Steven Sokol), Брайан К. Вест, Джон Тодд и Вильям Саффилл (William Suffill) (спасибо за мой самый первый VoIP-телефон, которым я пользуюсь до сих пор!). И всем тем, кого я обещал упомянуть в книге... спасибо!

И конечно, спасибо Джареду Смиту и Джиму Ван Меггелену за видение и понимание всей важности документации. Все это было бы невозможным без вас.

## Джаред Смит

Впервые я занялся Asterisk весной 2002 года. Незадолго до этого я устроился работать в компанию, занимающуюся исследованием рынка, и поехал в длительную командировку к удаленному центру телефонного обслуживания вместе с директором по информационным технологиям. По дороге домой мы долго говорили о нововведениях в телефонии, и он упомянул, что слышал о небольшом проекте телефонной связи с открытым исходным кодом под названием Asterisk. Через несколько месяцев мне удалось уговорить компанию купить комплект разработчика от Digium, и я начал упражняться с Asterisk в рабочее время.

За несколько месяцев я с головой ушел в сообщество разработчиков Asterisk. Я читал рассылки. Я перерыл архивы. Я висел в IRC-канале просто в надежде отыскать хоть какие-то крупицы сведений об Asterisk. Время шло, и я наконец приобрел достаточные знания, чтобы наладить Asterisk.

Вот тут началось настоящее веселье.

С помощью и при одобрении директора по информационным технологиям мы вынашивали планы перевести всю нашу инфраструктуру телефонной связи на Asterisk, включая офис компании и все удаленные центры телефонного обслуживания. В ходе работы мы натыкались на массу неведомых до тех пор проблем, и я начал подумывать о создании хорошего хранилища знаний по Asterisk. При этом нам удалось сделать несколько настоящих открытий, таких как объединение каналов IAX!

В конечном счете мы получили около сорока серверов Asterisk, рассредоточенных географически в разных местах, взаимодействующих друг с другом, образуя единую систему телефонной связи VoIP класса предприятия. Эта система в настоящее время обрабатывает примерно 1 млн минут звонков ежемесячно, обслуживает несколько сотен сотрудников, соединяется с 27 голосовыми линиями T1 и экономит для компании около $20 000 (США) каждый месяц на затратах на телефонную связь. Короче говоря, наш проект Asterisk имел ошеломительный успех!

Работая над реализацией этого проекта, где-то в IRC-каналах я познакомился с Лейфом. Мы поговорили о том, как могли бы помочь новым пользователям Asterisk и облегчить переход к этой системе, и решили настойчиво работать над планами по созданию более полной документации Asterisk. Я действительно хотел выпустить хорошую документацию в «бумажном» варианте, по сути, книгу, по которой новичок мог бы научиться основам Asterisk. Примерно в то же время чрезвычайно возросло количество новых пользователей в рассылках Asterisk и IRC-каналах, и мы почувствовали, что написание книги по Asterisk могло бы сильно улучшить соотношение количества полезной информации и «шелухи» в группе новостей. Так был рожден проект Asterisk Documentation Project! Все остальное, как говорится, уже история.

С тех пор я занимаюсь написанием документации Asterisk. Никогда не думал, что это настолько трудно, но в то же время полезно. (Мы с Лейфом и Джимом шутим, что, наверное, проще было бы написать исчерпывающий том под названием «Религия, контроль над оружием и суши», чем достаточно детально охватить все, что предлагает Asterisk!) То, что вы видите здесь, – прямой результат многих затягивавшихся до глубокой ночи посиделок и долгих выходных, проведенных на благо сообщества разработчиков Asterisk. Тем не менее это самое малое, что мы могли сделать, учитывая то, что Asterisk дала нам. Надеемся, эта книга вдохновит других членов сообщества разработчиков Asterisk на участие в доработке документации, внесении изменений и новых функций на благо всем.

Теперь время выразить благодарность.

Прежде всего хотел бы поблагодарить мою красавицу жену за те многие одинокие вечера, которые она провела, когда я корпел за клавиатурой. Я хотел бы, чтобы она знала, как я благодарен ей и как сильно ценю ее бесконечную поддержку. Также хочу сказать спасибо своим детям за то, что они никогда не дают мне забыть о самом важном в жизни. Я люблю вас!

Спасибо моим родителям за все, что они сделали, помогая мне расти, развиваться и учиться все эти годы. Вы – лучшие родители, о которых каждый может только мечтать.

Спасибо Дейву Карру (Dave Carr) и Майклу Ландбергу (Michael Lundberg) за то, что позволили мне изучать Asterisk в рабочее время. Работать с вами было настоящим удовольствием. Пусть удача улыбается вам и дарует успех во всех ваших начинаниях.

Спасибо Лейфу и Джиму за то, что выдерживали мои глупые шутки, мое упорство в желании делать все «правильно» и мой сумасшедший график. Спасибо за подстегивание и развитие моих писательских навыков. Мне действительно понравилось работать с вами. Надеюсь на сотрудничество в будущих проектах!

Спасибо Марку Спенсеру за непрекращающуюся поддержку, самоотверженность и дружбу. Вы были бесценным ресурсом для нашего начинания, и я искренне верю, что вы положили начало революции в мире телефонной связи. Вы всегда желанный гость в моем доме и за моим столом!

Спасибо всем остальным замечательным ребятам из Digium за помощь и поддержку. Мы особенно благодарны за вашу готовность помочь нам лучше понять код Asterisk и за предоставленное бесплатно оборудование, благодаря чему мы смогли более детально задокументировать комплект для разработчиков Asterisk (Asterisk Developer's Kit).

Спасибо Стивену Соколу, Стивену Критчфилду (Steven Critchfield), Олли И. Йоханссону и всем остальным, кто принимал участие в Asterisk Documentation Project и в создании данной книги! Мы не смогли бы ничего сделать без вашей помощи и советов.

# 1

# Революция в телефонии

*Для достижения цели не требуется иметь поддержку большинства, достаточно нескольких неистовых и неутомимых борцов, способных разжечь пламя в умах людей.*

– Самюэль Адамс

Мы являемся свидетелями невероятных революционных событий. Они ожидались уже давно, и теперь, когда процесс начался, ничто не в силах остановить его. Изменения охватили технологическую область, которая сильно отстала от всех остальных отраслей промышленности, объединенных общим названием hi-tech (от англ. high technology – высокая технология). Речь идет о телекоммуникациях, революцию в которых осуществил продукт с открытым исходным кодом для офисной телефонной станции с выходом в общую сеть (Private Branch eXchange, PBX) под названием *Asterisk*™.

Телекоммуникации – это, наверное, единственная из высокотехнологичных отраслей, которой не коснулась революция, связанная с появлением открытого исходного кода[1]. Основные производители в этой области по-прежнему создают необоснованно дорогие, несовместимые друг с другом системы, которые используют архаичное и запутанное программное обеспечение и впечатляющее своей инженерной мыслью, но безнадежно устаревшее оборудование.

Например, Business Communications Manager от компании Nortel каким-то чудом объединяет в себе кнопочный номеронабиратель 15-лет-

[1] До настоящего времени.

ней давности и ПК на базе процессора Celeron с частотой 1,2 ГГц[1]. Все это может стать вашим всего за $5000–15 000, не включая стоимость телефонных аппаратов. Если хочется получить какие-то действительно интересные функции, придется доплатить за универсальные приложения с ограниченной функциональностью и закрытым исходным кодом. Настройка? Забудьте о ней – она не входит в функционал системы. Технологии будущего и совместимость со стандартами? Подождите пару лет – над этим работают.

Все основные производители средств связи предлагают подобные продукты. Производители не хотят обеспечить вам возможность выбора или гибкость, а заинтересованы в том, чтобы потребитель был ограничен рамками жестко фиксированной функциональности их продуктов.

Система Asterisk вносит коренные изменения. С Asterisk никто не может диктовать, как должна работать телефонная система или какая технология должна использоваться. Выбирайте любую. Asterisk твердо следует идее совместимости со стандартами, позволяя при этом наслаждаться свободой создания собственных новшеств. Выбор только за вами, Asterisk не накладывает никаких ограничений.

Однако за такую невероятную гибкость приходится платить: Asterisk не назовешь системой, которую легко конфигурировать. И не потому, что она нелогична, запутанна или непонятна; напротив, она очень разумно сконструирована и удобна в применении. У человека, впервые увидевшего диалплан (рабочую среду) Asterisk и начинающего осознавать его возможности, просто загораются глаза. Но когда есть буквально тысячи способов достижения результата, естественно, процесс требует дополнительных усилий. Наверное, это можно сравнить с постройкой дома: все компоненты по отдельности просты и понятны, но человеку, чтобы выполнить такой проект, придется или a) обратиться за помощью к специалистам, или b) развить у себя необходимые навыки посредством обучения, практики и хорошего справочника по данной теме.

## VoIP: наведение мостов между традиционной и сетевой телефонией

Хотя передача голоса по IP-протоколу (Voice over IP, VoIP) часто рассматривается как своего рода бесплатная междугородняя телефонная связь, настоящая ценность VoIP в том, что с его помощью голос становится всего лишь обычным приложением в сети передачи данных.

---

[1] Не без удовольствия отметим, что Nortel наконец избавилась от Windows NT 4.0 и установленного Linux. С технической точки зрения идея хорошая, но довольно неожиданная, учитывая, что Nortel и Microsoft недавно объявили о партнерстве в разработке телекоммуникационных приложений уровня предприятия.

Кажется, мы забыли о том, что назначение телефона – позволить людям общаться. Это простая цель на самом деле, и мы должны иметь возможность реализовывать ее намного более гибко и творчески, чем это предлагается сейчас. Поскольку отрасль продемонстрировала нежелание стремиться к данной цели, решением задачи занялись энтузиасты.

Сложность состоит в том, что отрасль, которая практически не изменилась за последние сто лет, не проявляет особого интереса к этому и сейчас.

## Проект телефонной связи Zapata

Проект телефонной связи Zapata (Zapata Telephony Project) был основан Джимом Диксоном, инженером-консультантом по связи. Его вдохновило невероятное увеличение частот ЦП (центрального процессора), которое в компьютерной отрасли сейчас уже воспринимается как должное. Диксон считал, что при наличии плат, включающих только базовые электронные компоненты, необходимые для взаимодействия с телефонной сетью, можно было бы создать намного более экономичные системы телефонной связи. Дорогие компоненты не нужны, потому что вся цифровая обработка сигнала (Digital Signal Processing, DSP – ЦОС)[1] происходила бы в ЦП под управлением программного обеспечения. При этом нагрузка на ЦП сильно возросла бы, но Диксон был уверен, что низкая стоимость ЦП по сравнению с их производительностью делает их применение намного более привлекательным, чем использование ЦОС, и, что еще более важно, соотношение цена/производительность продолжало бы улучшаться с повышением мощности ЦП.

Как все мечтатели, Диксон верил, что эта идея откроется многим и ему просто надо подождать, пока кто-нибудь другой не реализует то, что он видел как очевидное усовершенствование. Но через несколько лет такие платы не только не были созданы, но, казалось, никто и не собирался ими заниматься. Тогда ему стало ясно, что если он хочет совершить революцию, то должен начинать ее самостоятельно. И родился проект телефонной связи Zapata.

Поскольку эта идея была настолько революционной и, несомненно, вызвала бы большой резонанс в отрасли, я обратился к мотивам революции в Мексике и назвал технологию и организацию по имени известного мексиканского революционера Эмилиано Запата (Emiliano

[1] Аббревиатура DSP также расшифровывается как Digital Signal Processor (процессор цифровой обработки сигналов). Это устройство (обычно чип), способное обрабатывать и изменять различные сигналы. В сети телефонной связи процессоры ЦОС преимущественно отвечают за кодировку, декодировку и перекодировку аудиоинформации. Для этого может требоваться выполнять большой объем вычислений.

Zapata). Для платы я выбрал имя tormenta, что по-испански означает «буря», и обычно подразумевается сильная буря, например ураган или нечто подобное[1].

Возможно, нам следовало бы называть себя астеритянами. Мы, конечно, в долгу перед Джимом Диксоном и за то, что он все это придумал, и за то, что довел дело до конца, но прежде всего – за предоставление результатов своего труда сообществу разработчиков продуктов с открытым исходным кодом. Благодаря Джиму появилось ядро коммутируемой телефонной сети общего пользования (Public Switched Telephone Network, PSTN) Asterisk.

## Для широкомасштабных изменений необходима гибкая технология

Самая успешная в мире малая АТС имеет конструктивное ограничение, об устранении которого пользователи умоляют вот уже в течение 15 лет: при определении того, сколько раз прозвонит телефон, прежде чем вызов будет перенаправлен на голосовую почту, предоставляется возможность выбрать 2, 3, 4, 6 или 10 звонков. Знаете ли вы, сколько человек просили о внесении возможности выбора пяти звонков? Казалось бы, требуется внести простое изменение, но, сколько бы ни просили пользователи, производители не могут понять, что это действительно является проблемой. «Она так работает, – отвечают они, – и пользователям надо просто смириться с этим».

Другой подобный пример: имя в телефонной книге может быть длиной не более семи символов[2]. В конце 1980-х, когда эта система разрабатывалась, оперативная память была очень дорогой и хранение семи символов для десятков телефонных аппаратов означало гигантские расходы на оборудование. А какое этому оправдание может быть сегодня? Его нет. Планируется ли изменить ситуацию? Вряд ли, вопрос даже не признан проблемой официально.

Это всего два примера, а отрасль изобилует ими.

Мы рассмотрели одну систему, но реальное положение дел таково, что недостатки есть во всех существующих офисных АТС. Неважно, насколько богатую функциональность предлагает телефонная станция, – учесть все и предвидеть изобретательность пользователя невозможно. Нескольким пользователям может потребоваться маленькая необычная возможность, о которой группа разработки или не подумала или

---

[1] Джим Диксон «The History of Zapata Telephony and How It Relates to the Asterisk PBX» (*http://www.asteriskdocs.org/modules/tinycontent/index.php?id=10*).

[2] Если необходимо ввести имя Елизавета, придется придумать что-нибудь наподобие Елизавт, или Елизаве, или, скажем, Лизавет. Есть, конечно, приемлемый вариант – Лиза, но суть проблемы ясна.

решила не заниматься ею из-за неоправданности затрат на ее разработку, а поскольку код системы закрыт, пользователи не смогут самостоятельно реализовать необходимую функциональность.

Если бы всякого рода правила и коммерческие интересы сдерживали развитие Интернета, он никогда не получил бы такого широкого распространения. Открытость Интернета означает, что каждый желающий может поучаствовать в его разработке. В результате совместного труда десятков тысяч умов получен продукт, который не мог бы выйти из стен ни одной корпорации.

Как и для многих проектов с открытым исходным кодом, таких как Linux и Интернет, импульсом к разработке Asterisk были мечты тех, кто знал, что должно существовать нечто большее, чем предлагается в данной отрасли. Сила сообщества в том, что его составляют не служащие, решающие поставленные перед ними конкретные задачи, а люди из всевозможных областей деятельности с совершенно различным опытом и разным пониманием гибкости и открытости. Эти люди знали: если суметь выделить лучшее, что есть в разных АТС, в отдельные компоненты, которые можно различным образом соединять между собой, подобно блокам LEGO, начнут появляться идеи, которые не прошли бы традиционный в корпорациях процесс анализа рисков. До тех пор пока ни у кого нет полной картины того, как все должно выглядеть, недостатка во мнениях и идеях нет[1].

Многие люди, впервые сталкиваясь с Asterisk, считают ее незаконченным проектом. Наверное, их можно сравнить с посетителями изостудии, ожидающими увидеть здесь подписанные и пронумерованные репродукции. Часто они разочаровываются, узнав, что Asterisk – это ожидающие их чистый холст, тюбики с краской и новые кисти[2].

Даже на этом раннем этапе, на котором уже удалось достичь успеха, проектом Asterisk занимаются больше мастеров, чем любой другой офисной АТС. В большинстве компаний-производителей над каким-либо продуктом трудятся лишь несколько разработчиков; в разработке Asterisk участвуют очень много людей. Для обслуживания большинства коммерческих АТС во всем мире найдется лишь несколько десятков настоящих экспертов; в случае с Asterisk их сотни.

---

[1] В версии Asterisk 1.4, по сравнению с версией 1.2, было внесено более 4000 обновлений в код хранилища SVN.

[2] Но не стоит огорчаться. Появилось несколько проектов, которые помогут упростить внедрение Asterisk. На данный момент наиболее популярным и известным из них является trixbox (*http://www.trixbox.org*). Имея старый ПК (или виртуальную машину VMware), с помощью графического интерфейса trixbox можно настроить офисную АТС, просто ответив на несколько вопросов в процессе автоматической установки. Это не поможет научиться работать с Asterisk, потому что здесь пользователь не участвует в конфигурации платформы или диалплана, но так вы получите рабочую АТС намного быстрее, чем если бы делали это преимущественно вручную, как описывается в данной книге.

Глубина и широта экспертных знаний, вложенных в этот продукт, не имеет аналогов в телефонной отрасли. Asterisk имеет преданных поклонников среди «стариков» из Telco, бывших свидетелями расцвета телефонных аппаратов с дисковыми номеронабирателями, сотрудников крупных телекоммуникационных компаний, которые помнят времена, когда голосовая почта была самой модной новейшей технологией, и специалистов по передаче данных, помогавших создавать Интернет. Все эти люди верят в одно – телекоммуникационная промышленность нуждается в *надлежащих* революционных изменениях[1].

Asterisk – это катализатор.

## Asterisk: офисная АТС, создаваемая хакерами

Телекоммуникационные компании, которые решили игнорировать Asterisk, поступают рискованно. Предоставляемая Asterisk гибкость обеспечивает возможности, о которых лучшие коммерческие системы могут только мечтать, потому что Asterisk – это АТС, созданная хакерами.

Если вас попросят не применять слово *«хакер»*, не обращайте внимания. Этот термин не является собственностью средств массовой информации. Они присвоили его и исказили значение, называя так «злонамеренных взломщиков». Пришло время восстановить справедливость. Хакеры разработали механизм передачи данных по сети, то есть Интернет. Хакеры создали Apple Macintosh и операционную систему UNIX. Хакеры работают и над телефонной системой будущего. Не надо пугаться, все они отличные парни и смогут построить систему, намного более безопасную, чем все существующие сегодня. Они не станут создавать ненадежные и легко поддающиеся взлому средства безопасности для закрытых систем, хакеры смогут быстро реагировать на изменение тенденций в обеспечении безопасности и настраивать телефонную систему соответственно политике корпорации и наилучшей практике отрасли.

Как и другие системы с открытым исходным кодом, Asterisk сможет развиться в намного более безопасную платформу, чем любая коммерческая система, не вопреки своим хакерским корням, а, скорее, благодаря им.

## Asterisk: офисная АТС, создаваемая профессионалами

Никогда за всю историю телекоммуникаций не существовало системы, настолько отвечающей нуждам бизнеса в любой ценовой категории. Asterisk – технология, предоставляющая новые возможности, и, как

[1] Революцию в телекоммуникациях предсказывали еще до краха старых технологий; время покажет, насколько эффективно было революционное введение *открытого кода*.

это было с Linux, скоро вряд ли можно будет найти предприятие, на котором не использовалась бы одна из версий Asterisk, хотя бы отчасти, где-то в сети, для решения проблем, которые способна разрешить только Asterisk.

Однако похоже, что это признание произойдет быстрее, чем было в случае с Linux, по ряду причин:

- Linux уже проложил путь к признанию открытого исходного кода. Asterisk идет по проторенной дороге.
- Телефония находится в бедственном положении, ни один из крупных игроков на этом рынке не является лидером. Asterisk же представляется убедительным, реалистичным и впечатляющим проектом.
- Конечные пользователи уже сыты по горло несовместимыми системами с ограниченной функциональностью и ужасной поддержкой. Asterisk решает первые две проблемы – предприниматели и сообщество обеспечат последнее.

## Сообщество разработчиков Asterisk

Одна из неоспоримых сильных сторон системы телефонии Asterisk – сообщество энтузиастов, разработавших и поддерживающих его, руководит которым Марк Спенсер, основатель компании Digium. Сообщество остро осознает культурную значимость Asterisk и с упоением смотрит в будущее.

Самый значимый результат деятельности сообщества разработчиков Asterisk – объединение профессионалов из разных областей: телекоммуникаций, сетевых и информационных технологий, – которые прониклись любовью к этому феномену. Несмотря на традиционную конфронтацию между этими отраслями, в сообществе Asterisk эти специалисты восхищаются способностями друг друга. Важность такого сотрудничества нельзя недооценивать.

Тем не менее для реализации мечты об Asterisk сообщество должно расширяться. На данный момент одной из основных проблем является быстрый приток новых пользователей. Действительные члены сообщества, положившие начало тому, что называется Asterisk, в общем, рады новым участникам, но обеспокоены тем, что им приходится сталкиваться с вопросами, ответы на которые можно найти самостоятельно, если потратить немного времени на поиски и эксперименты.

Очевидно, что все новые участники не могут быть одинаковыми. Кто-то будет рад проводить часы, экспериментируя и читая различные блоги, описывающие чьи-то злоключения и мучения. Но многие из тех, кто проникся этой технологией, совершенно не заинтересованы вести такие поиски. Они хотят иметь простое и понятное пошаговое руководство, которое введет их в курс дела, сопровождаемое некоторыми полезными

примерами, где описаны лучшие методы реализации обычной функциональности (такой, как голосовая почта, автосекретарь и пр.).

Эксперты сообщества, которые (совершенно правильно) считают Asterisk своего рода языком разработки веб-приложений, не приемлют такого подхода. Для них очевидно, что постичь все тонкости Asterisk можно, только полностью погрузившись в нее. Разве кому-нибудь придет в голову просить пошаговое руководство по программированию и надеяться научиться по нему всему, что предлагает язык программирования?

Конечно, нет единого подхода, который подошел бы всем. Asterisk абсолютно ни на что не похожа и требует совершенно иного типа мышления. Знакомясь с сообществом, помните, что в нем собрались люди с разными навыками и характерами. Некоторые из этих парней не отличаются особой сдержанностью при общении с новичками, но это лишь потому, что они очень трепетно относятся к предмету, а не потому, что не рады вам.

## Рассылки по Asterisk

Как и в любом другом сообществе, существуют сайты, где члены сообщества разработчиков Asterisk собираются для обсуждения вопросов, вызывающих всеобщий интерес. Список рассылок можно найти по адресу *http:// lists.digium.com*. Самыми значительными из них на настоящий момент являются вот эти четыре:

*Asterisk-Biz*

В этой рассылке представлено все, что касается коммерческой деятельности, связанной с Asterisk. Если вы продаете что-то, связанное с Asterisk, делайте это здесь. Если хотите купить услугу или продукт Asterisk, пишите сюда.

*Asterisk-Dev*

Здесь обитают разработчики Asterisk. Назначение этой рассылки – обсуждение разработки программы Asterisk, и его участники энергично отстаивают это. Ожидайте шквала негодования, если отправили в эту рассылку что-то, не относящееся непосредственно к программированию или разработке кодовой базы Asterisk. Общие вопросы по написанию кода (такие, как взаимодействие с AGI или AMI) следует направлять в рассылку Asterisk-Users.

Рассылка Asterisk-Dev не является средством технической поддержки пользователей! Если просмотреть архивы рассылок, можно увидеть, что это правило строго соблюдается. В Asterisk-Dev обсуждается разработка Asterisk, а вопросы о взаимодействии внешних программ через AGI или AMI следует направлять в рассылку Asterisk-Users.

*Asterisk-Users*

Это рассылка для пользователей Asterisk с более чем десятью тысячами подписчиков. В ней формируется несколько сотен сообщений в день. Сюда можно обратиться за помощью, но все-таки присылайте свои вопросы только после того, как попробовали и не смогли найти ответ самостоятельно.

*Asterisk-BSD*

Здесь высказываются члены сообщества, реализующие Asterisk под FreeBSD (и другие диалекты BSD).

## Википедия об Asterisk

Раздел Википедии по Asterisk (который существует по большей части благодаря неутомимым усилиям Джеймса Томпсона (James Thompson) – *спасибо тебе, Джеймс!*) – источник просвещения и путаницы. Управляемое сообществом хранилище знаний по VoIP (*http://www.voip-info.org*) – это просто кладезь интереснейшей, содержательной и часто противоречивой информации по многим вопросам, Asterisk является лишь одним из них.

Поскольку документация по Asterisk составляет основную массу информации на данном веб-сайте[1] и, вероятно, на нем содержится больше сведений об Asterisk, чем во всех остальных источниках, вместе взятых (за исключением архивов рассылок), обычно этот веб-сайт указывают как *основной* ресурс данных об Asterisk[2].

## Каналы IRC

Сообщество разработчиков Asterisk поддерживает каналы ретрансляции интернет-чатов (Internet Relay Chat, IRC) на сайте *irc.freenode.net*. Самыми активными каналами являются *#asterisk* и *#asterisk-dev*[3]. В целях защиты от спама теперь на обоих каналах требуется регистрация[4].

## Группы пользователей Asterisk

На многих сайтах по всему миру одинокие пользователи Asterisk начинают осознавать, что в их городах есть и другие люди, разделяющие их пристрастие. Группы пользователей Asterisk (Asterisk User Groups,

---

[1] По последним подсчетам более 30%.

[2] *http://ru.wikipedia.org/wiki/Asterisk* – русскоязычный раздел Википедии, посвященный Asterisk. – *Примеч. науч. ред.*

[3] Канал *#asterisk-dev* посвящен изменениям в базовом коде Asterisk и не предоставляет технической поддержки пользователям. Вопросы, касающиеся программирования внешних приложений, которые взаимодействуют с Asterisk через AGI или AMI, должны направляться в *#asterisk*.

[4] `/msg nickserv` помогает при соединении с сервером через ваш любимый IRC-клиент.

AUGs) возникают повсюду. Хотя они никак официально не взаимосвязаны, обычно они дают ссылки на сайты друг друга и всегда рады любым новым членам. Введите в Google поисковые слова «Asterisk User Group», чтобы найти группу в своем регионе[1].

## Проект создания документации Asterisk

Проект создания документации Asterisk (Asterisk Documentation Project) начали осуществлять Лейф Мадсен и Джаред Смит, но в нем участвовали и другие члены сообщества.

Цель проекта – создание структурированного хранилища письменных источников по Asterisk. В противоположность гибкой и случайной природе Википедии, проект Docs направлен на формирование более узкоспециализированного подхода к различным связанным с Asterisk вопросам.

В рамках проекта Asterisk Docs, нацеленного на то, чтобы сделать документацию доступной в Сети, данная книга представлена на вебсайте *http://www.asteriskdocs.org* по лицензии Creative Commons.

# Экономическое обоснование

Сегодня практически невозможно найти предприятие, которому не приходилось бы перестраиваться каждые несколько лет. Так же сложно найти компанию, которая может позволить себе заменять свою инфраструктуру связи при каждой смене курса. Современному бизнесу необходима предельная гибкость во всех используемых технологиях, включая телекоммуникацию.

В своей книге «Crossing the Chasm» (HarperBusiness) Джеффри Мур (Geoffrey Moore) говорит: «Идея того, что ценность системы будет раскрываться постепенно и не будет известна на момент установки, подразумевает, в свою очередь, что гибкость и приспособляемость продукта, так же как и постоянное обслуживание клиентов, должны быть основными критериями оценки при покупке любой системы».

В частности, это означает, что истинная ценность технологии порой бывает неясна вплоть до ее развертывания.

Теперь вы можете оценить привлекательность системы, в основу которой положена концепция открытости и постоянного обновления.

# Об этой книге

Итак, с чего начнем? Об Asterisk можно рассказать столько, что одной книги не хватит. Мы не собираемся здесь вдаваться во все тонкости, просто рассмотрим основы.

---

[1] Русскоязычные ресурсы по Asterisk: *http://asteriskpbx.ru/*, *http://www.asterisk-support.ru/*, *http://www.asteriskforum.ru/* – *Примеч. науч. ред.*

В главе 2 обсуждаются некоторые вопросы проектирования, которые следует учитывать при планировании телекоммуникационных систем. Бóльшую часть данного материала можно пропустить и перейти прямо к установке, но эти идеи важно понимать тем, кто планирует вводить в эксплуатацию систему Asterisk.

Глава 3 посвящена тому, как получить, откомпилировать и установить Asterisk. В главе 4 речь идет об исходной конфигурации Asterisk. Здесь рассматриваются важные конфигурационные файлы, которые должны иметься для определения каналов и функций, доступных в конкретной системе. Этот материал подготовит почву для главы 5, где представлено сердце Asterisk – диалплан. Глава 6 ознакомит вас с некоторыми более сложными концепциями диалплана.

В главе 7 мы отдохнем от Asterisk и обсудим некоторые наиболее важные технологии, используемые в PSTN. В главе 8, посвященной действующим системам телефонии, мы, естественно, обсудим технологию передачи голоса по IP-протоколу.

В главе 9 представлен один из наиболее удивительных компонентов, шлюзовой интерфейс Asterisk (Asterisk Gateway Interface, AGI). Используя языки программирования Perl, PHP и Python, мы продемонстрируем, как можно использовать внешние программы для добавления в офисную АТС практически безграничных функциональных возможностей. В главе 14 кратко рассматриваются невероятные возможности и функции, составляющие феномен Asterisk. В завершение, глава 15 «заглядывает вперед», предсказывая будущее, в котором телефония с открытым исходным кодом полностью преображает отрасль, отчаянно нуждающуюся в революционных изменениях. Также в книге можно найти массу справочной информации, которая приводится в пяти приложениях.

Эта книга только закладывает основы, но на базе почерпнутых из нее знаний вы сможете прийти к пониманию концепции Asterisk, и кто знает, что вы тогда создадите.

# 2

# Подготовка системы к установке Asterisk

*Я очень рано понял, что когда-нибудь, где-то там за горизонтом, в некотором «идеальном» будущем, все необходимые функции обработки данных будут выполняться централизованно внутри компьютеров, что приведет к значительному удешевлению, а в некоторых случаях обесцениванию внешнего оборудования, необходимого для соединения с телекоммуникационными интерфейсами.*

– Джим Диксон «The History of Zapata Telephony and How It Relates to the Asterisk PBX»

Вам, должно быть, уже не терпится настроить собственную систему Asterisk. Если вы планируете создать любительскую систему, то, пожалуй, можете перейти сразу к следующей главе и начать установку. Однако, если Asterisk развертывается для решения ответственных задач, необходимо сказать несколько слов о среде, в которой она будет выполняться. Будьте уверены, Asterisk – очень гибкое ПО и успешно устанавливается практически на любую платформу Linux, а также на несколько не-Linux платформ[1]. Но в данной главе обсуждаются вопро-

[1] Есть опыт успешной компиляции и выполнения Asterisk на платах WRAP, маршрутизаторах Linksys WRT54G, системах Soekris, процессорах Pentium 100, PDA, Apple Mac, Sun SPARC, портативных компьютерах и многих других. Конечно, совершенно другой вопрос, *захотите ли* вы вводить в эксплуатацию такую систему. (Вообще говоря, реализация AstLinux, выполненная Кристианом Келхофнером, действительно замечательно осуществляется на плате Soekris 4801. К этому вопросу стоит вернуться после ознакомления с основами Asterisk. Загляните на сайт *http://www.astlinux.org*.)

сы, знание ответов на которые вооружит вас пониманием того, в каком операционном окружении Asterisk будет действительно эффективно функционировать, и поможет создать надежную, хорошо спроектированную систему.

С точки зрения требований к ресурсам Asterisk подобна встроенным системам реального времени преимущественно тем, что она должна иметь приоритетный доступ к процессору и системным шинам. Поэтому крайне важно, чтобы все остальные функции системы, не связанные напрямую с задачами Asterisk по обработке вызовов, если таковые вообще выполняются, должны выполняться с более низким приоритетом. Для небольших и любительских систем это может и не представлять особой проблемы. Однако для высокопроизводительных систем недостаточная производительность будет вызывать проблемы с качеством аудиосигнала, получаемого пользователем, часто в виде эха, помех и т. п. Примерно так ведут себя устройства мобильной связи при выходе из зоны обслуживания, но здесь причина этих проблем другая. По мере увеличения нагрузки на систему будут возрастать сложности с обслуживанием соединений. Для офисной АТС подобная ситуация – настоящая катастрофа, поэтому в процессе выбора платформы требования к производительности должны быть решающим критерием.

В табл. 2.1 представлены некоторые самые основные рекомендации к планированию системы. В следующем разделе подробно рассматриваются различные вопросы проектирования и реализации, связанные с производительностью системы.

Размер системы Asterisk на самом деле определяется не количеством пользователей или телефонных аппаратов, а, скорее, количеством одновременных вызовов, которые система должна будет поддерживать. Эти цифры очень приблизительны, поэтому экспериментируйте и выбирайте наиболее подходящий для себя вариант.

*Таблица 2.1. Рекомендации по выбору технических характеристик системы*

| **Назначение** | **Количество каналов** | **Рекомендуемые минимальные параметры** |
|---|---|---|
| Любительская система | Не более 5 | 400 МГц ×86, 256 Мб оперативной памяти |
| SOHO-система (малый офис и дом – менее трех линий и пяти телефонных аппаратов) | От 5 до 10 | 1 ГГц ×86, 512 Мб оперативной памяти |
| Малая бизнес-система | До 25 | 3 ГГц ×86, 1 Гб оперативной памяти |
| Средняя или большая система | Более 25 | Два ЦП, возможно также несколько серверов в распределенной архитектуре |

### Результаты нагрузочного тестирования

Джошуа Колп (Joshua Colp) смог получить результаты, приведенные в табл. 2.2, используя процессор AMD Athlon64 X2 4200+ с 1 Гб оперативной памяти и жестким диском SATA емкостью 80 Гб и проводя тестирования по стандартному сценарию в приложении SIPp: простое установление соединения, воспроизведение аудиофайла (приложение `Playback()`) и некоторый небольшой период ожидания (`Wait()`). Обратите внимание на существенное снижение использования ресурсов ЦП при чтении данных из оперативной памяти по сравнению с чтением с жесткого диска. Это можно истолковать так, что ЦП ожидает данные, подлежащие обработке, перед передачей их в запрашивающий канал. Однако это всего лишь простой тест, и он никоим образом не отражает, какое количество вызовов сможет обрабатывать ваша система. Определить количество одновременных вызовов, которое может быть обработано при использовании конкретного диалплана и сочетания приложений, можно, только проведя нагрузочное тестирование системы.

*Таблица 2.2. Пример результатов тестирования для стандартного сценария SIPp, использующего простые приложения Wait( ) и Playback( ); SIPp отражает обратный медиа-поток Asterisk*

| Количество одновременных вызовов | 330 | 330 | 550 |
|---|---|---|---|
| Использование ЦП, % | 149 | 14,8 | 57,6 |
| Средняя нагрузка | 49 | 25 | 60 |
| Запоминающее устройство | Жесткий диск | ОЗУ | ОЗУ |

Для больших установок Asterisk функциональность обычно распределяют между несколькими серверами. Один или более центральных модулей будут заниматься обработкой вызовов; их дополнят один или более вспомогательных серверов, обслуживающих периферийные устройства (такие, как система баз данных, система голосовой почты, система конференц-связи, система управления, веб-интерфейс, межсетевой экран и т. д.). Asterisk, как и многие Linux-системы, может расширяться с ростом требований к ней: малая система, которая прекрасно справлялась со всеми задачами по обработке вызовов и обслуживанию периферийных устройств, может быть распределена между несколькими серверами, когда требования возрастут и превысят ее текущие

возможности. Гибкость – основная причина, по которой Asterisk исключительно рентабельна для быстро растущего бизнеса; для нее не существует эффективного максимального или минимального размера, который следует учитывать при составлении сметы на покупку. Хотя масштабируемость свойственна большинству телефонных систем, до сих пор нам не приходилось слышать о системе, которая была бы настолько же гибкой, как Asterisk. Однако стоит отметить, что задача по проектированию распределенных систем Asterisk не по зубам новичку в Asterisk.

Тем, кто намерен настраивать распределенную систему Asterisk, рекомендуется изучить протокол DUNDi, архитектуру реального времени Asterisk (Asterisk Realtime Architecture, ARA), func_odbc и другие имеющиеся в распоряжении инструменты для работы с базами данных. Это поможет научиться извлекать из логики диалплана необходимые вашей системе данные, которые будут использоваться системой Asterisk. Это делает возможным существование универсального множества логик диалплана, которое может использоваться во множестве блоков. Тогда для масштабирования системы необходимо просто ввести в нее дополнительные блоки. Однако вопросы масштабирования выходят далеко за рамки данной книги, оставим это как упражнение для читателя. Некоторые инструменты, которые могут использоваться для масштабирования, рассмотрены в главе 12.

# Выбор серверного оборудования

Задача по выбору сервера проста и сложна одновременно. Проста потому, что на самом деле подойдет любая платформа на базе ×86, а сложна потому, что гарантированное обеспечение необходимой производительности системы будет зависеть от того, насколько тщательно спроектирована платформа. При выборе оборудования следует внимательно рассмотреть конструкцию системы в целом и то, какие функциональные возможности требуется поддерживать. Это поможет определить требования к ЦП, системной плате и блоку питания. Те, кто просто настраивает свою первую систему Asterisk с целью научиться это делать, могут спокойно проигнорировать информацию, приведенную в данном разделе. Однако при построении полноценной системы, пригодной к практическому применению, рассматриваемые здесь вопросы требуют проработки.

## Вопросы производительности

При выборе оборудования для установки Asterisk главным соображением является то, насколько мощной должна быть полученная система. Это непростой вопрос, поскольку большую роль в данном случае

играет то, как будет использоваться система. Такого понятия, как модель управления производительностью Asterisk, не существует, поэтому, чтобы принять разумное решение о необходимых ресурсах, следует определить, как Asterisk будет использовать систему. Должны быть учтены следующие факторы:

*Максимальное число одновременных соединений, которое система должна будет поддерживать*

*Каждое* соединение будет увеличивать нагрузку на систему.

*Доля трафика в процентном выражении, который потребует интенсивной работы процессора для ЦОС кодеками, использующими алгоритмы сжатия (такими, как G.729 и GSM)*

Работа по цифровой обработке сигнала (Digital Signal Processing, DSP), которую Asterisk осуществляет на программном уровне, может иметь огромное влияние на то, какое количество одновременных вызовов она будет поддерживать. Система, которая успешно обрабатывает 50 одновременных вызовов G.711, может потерпеть фиаско при запросе на одновременную обработку 10 каналов со сжатием, кодированных G.729. Мы подробнее поговорим о G.729, GSM, G.711 и многих других кодеках в главе 8.

*Будет ли обеспечиваться конференц-связь и какая интенсивность общения предполагается*

Будет ли система использоваться интенсивно? Конференц-связь требует, чтобы система преобразовывала и добавляла каждый отдельный входной аудиопоток во множество выходных потоков. Смешение нескольких аудиопотоков в масштабе времени, близком к реальному, может создавать существенную нагрузку на ЦП.

*Эхоподавление*

Эхоподавление может потребоваться при любом вызове, в котором задействован интерфейс коммутируемой телефонной сети общего пользования (Public Switched Telephone Network, PSTN). Поскольку эхоподавление является математической функцией, чем больше системе приходится его выполнять, тем выше будет нагрузка на ЦП[1]. Не пугайтесь. Эхоподавление – это еще одна тема для главы 8.

*Логика разработки сценариев диалплана*

Передача Asterisk управления вызовами внешней программе всегда ведет к снижению производительности. Логика максимально должна быть реализована внутри диалплана. Если используются внешние сценарии, основными критериями при их создании должны быть производительность и эффективность.

Как именно эти факторы влияют на производительность, сказать наверняка сложно. Эффект от каждого из них описан в общем, но точного

[1] Частота ЦП примерно 30 МГц на канал.

количественного выражения еще нет. Отчасти это объясняется тем, что воздействие каждого компонента системы зависит от множества величин, таких как тактовая частота ЦП, набор микросхем системной платы и общее качество, общий информационный трафик системы, оптимизации ядра Linux, сетевой трафик, количество и тип интерфейсов PSTN и трафик PSTN, не говоря уже о сервисах, не относящихся к Asterisk, которые выполняются системой параллельно. Рассмотрим влияние некоторых ключевых факторов:

*Кодеки и перекодировка*

Попросту говоря, *кодек* (сокращение от кодер/декодер, или алгоритм уплотнения/разуплотнения данных) – это набор математических правил, который определяет, как будет выполняться оцифровка аналогового сигнала. Кодеки различаются, главным образом, предлагаемыми ими уровнями сжатия и качеством воспроизведения звука. Вообще говоря, чем бóльшая степень сжатия требуется, тем больше ЦОС должно быть выполнено при кодировании или декодировании сигнала. Следовательно, кодеки без сжатия являются менее тяжелыми для ЦП (но требуют более широкой полосы пропускания сети). Выбор кодека должен быть компромиссом между пропускной способностью и загруженностью процессора.

*Центральный процессор (и модуль обработки операций с плавающей точкой)*

ЦП состоит из нескольких компонентов, один из них – модуль обработки операций с плавающей точкой (Floating Point Unit, FPU). От производительности ЦП в сочетании с эффективностью блока FPU во многом зависит, какое количество одновременных соединений сможет эффективно поддерживать система. В следующем разделе («Выбор процессора») даются некоторые общие рекомендации по выбору ЦП соответственно нуждам конкретной системы.

*Другие процессы, параллельно выполняющиеся в системе*

Linux подобна операционной системе UNIX, то есть является многозадачной системой, которая может выполнять несколько разных процессов одновременно. Проблема возникает, когда один из этих процессов (такой, как Asterisk) требует от системы очень высокой быстроты реагирования. По умолчанию Linux распределяет все ресурсы между приложениями, запрашивающими их, поровну. Если установлена система, включающая множество разных серверных приложений, каждое из них получит свою равную долю времени ЦП. Поскольку системе Asterisk необходим частый высокоприоритетный доступ к ЦП, для обеспечения ее сосуществования с другими приложениями система требует специальной оптимизации. Главным образом, подразумевается назначение приоритетов различным приложениям в системе и внимательное отношение к тому, какие сервисы устанавливаются.

*Оптимизации ядра*

Очень немногие дистрибутивы Linux предлагают по умолчанию ядро, оптимизированное для выполнения одного конкретного приложения, поэтому здесь необходимо слегка потрудиться. Какой бы дистрибутив ни был выбран, как минимум, придется скачать и скомпилировать на своей платформе свежую версию ядра Linux (которую можно найти по адресу *http://www.kernel.org)*. Также можно найти патчи, которые обеспечат повышение производительности, но считаются неофициальными дополнениями к официально поддерживаемому ядру.

*Время ожидания запроса на прерывание*

Время ожидания запроса на прерывание (Interrupt request, IRQ) – это, по сути, задержка между моментом, когда периферийная плата (такая, как интерфейсная плата для телефонии) запрашивает ЦП остановить выполняемый процесс, и моментом, когда ЦП фактически отреагирует и будет готов обрабатывать задачу. Периферийные устройства Asterisk (особенно платы Zaptel) чрезвычайно требовательны ко времени ожидания IRQ. Причиной этому является не столько несовершенство плат, сколько принцип работы программного механизма временного уплотнения (TDM). Если мы буферизируем данные мультиплексора с временным уплотнением (TDM) и посылаем их на шину большим пакетом, это может быть более эффективным для системы, но обусловит задержку между моментом поступления аудиоданных на плату и доставкой их в ЦП. Это делает обработку данных TDM в масштабе реального времени практически невозможной. При проектировании Zaptel было принято, что отправка данных каждую 1 мс будет наилучшим компромиссным решением. Но в результате этого возникает побочный эффект – любая плата в системе, использующая интерфейс Zaptel, будет посылать в систему запрос на обработку прерывания каждую миллисекунду. Это было характерно для старых системных плат, но на данный момент уже почти перестало быть причиной для беспокойства.

В Linux всегда существовала проблема недостаточно быстрого обслуживания IRQ; это доставляло немало неприятностей разработчикам приложений для работы с аудиоданными, поэтому было создано несколько патчей для устранения этого недостатка. До сих пор нет единого мнения по поводу того, как включать эти патчи в ядро Linux.

*Версия ядра*

Asterisk официально поддерживается Linux версии 2.6.

*Дистрибутив Linux*

Дистрибутивов Linux много, и они разнообразны. В следующей главе мы обсудим проблему выбора дистрибутива Linux и то, как получить и установить и Linux, и Asterisk.

## Выбор процессора

Поскольку требования, предъявляемые Asterisk к производительности, главным образом, обусловлены большим объемом производимых математических вычислений, естественным будет выбор процессора с мощным FPU. Для осуществляемой Asterisk обработки сигналов от ЦП может потребоваться проведение громадного числа сложных математических вычислений. Эффективность, с которой выполняются эти задачи, будет определяться мощностью FPU процессора.

Назвать лучший процессор для Asterisk в этой книге означало бы бросить вызов закону Мура. Даже за то время, которое пройдет между написанием и публикацией книги, скорости процессоров существенно возрастут, так же как и поддержка Asterisk для различных архитектур. Несомненно, это хорошо, но по этой причине советы по данной теме являются абсолютно неблагодарным занятием. Естественно, чем мощнее FPU, тем больше одновременных задач по ЦОС сможет выполнять Asterisk. Это основной принцип. При выборе процессора исходная тактовая частота – только часть уравнения. То, насколько хорошо он справляется с операциями с плавающей точкой, будет основным определяющим фактором, поскольку операции по ЦОС в Asterisk будут предъявлять высокие требования именно к этому процессу.

ЦП производства компаний Intel и AMD имеют мощные FPU. От чипов текущего поколения любого из данных производителей можно ожидать хорошей производительности[1].

Сам собой напрашивается вывод, что необходимо выбирать самый мощный процессор из тех, которые позволяет ваш бюджет. Однако не торопитесь покупать самый дорогой из доступных процессоров. Помните о требованиях своей системы. В конце концов, болид «Формулы-1» Ferrari совершенно неуместен в мегаполисе с его многокилометровыми пробками в часы пик. Более медленные ЦП часто слабее нагреваются, и, таким образом, используя их, можно построить систему Asterisk для небольшого офиса с меньшим энергопотреблением, без вентиляторов, которая, возможно, смогла бы работать в условиях повышенной запыленности.

Чтобы ввести некий критерий, исходя из которого можно принимать решения о платформе, мы решили определить три типа систем Asterisk: малая, средняя и большая.

---

[1] Найти самую свежую информацию о том, какой из ЦП лидирует в гонке производительности, можно на сайтах Tom's Hardware (*http://www.tomshardware.com*) или AnandTech (*http://www.anandtech.com*), где представлена масса информации как о современных, так и об устаревших ЦП, системных платах и наборах микросхем.

## Малый тип систем

Требования Asterisk к производительности для малых систем (до 10 телефонов) ничем не отличаются от всех остальных, но, как правило, возможности современных процессоров позволяют справиться с нагрузкой, типичной для таких систем.

Если малая система создается на базе случайно подвернувшихся под руку старых компонентов, следует ожидать, что уровень производительности будет ниже, чем у более мощных машин, и что для нее снижение рабочих характеристик будет наблюдаться при значительно меньших нагрузках. Любительские системы могут прекрасно работать на маломощном оборудовании, но добиться этого сможет только эксперт в вопросах настройки производительности Linux[1].

Если система Asterisk настраивается в целях обучения, построить полнофункциональную платформу можно, используя относительно маломощный процессор. Авторы данной книги выполняли настройку нескольких лабораторных систем Asterisk с использованием процессоров Celeron с частотами от 433 до 700 МГц, но рабочая нагрузка таких систем минимальна (не более двух одновременных вызовов).

### AstLinux и Asterisk на OpenWRT

Те, кто действительно прекрасно себя чувствует, работая с Linux на встроенных платформах, несомненно, захотят присоединиться к рассылке AstLinux и опробовать творение Кристиана Кайлхофнера (Kristian Kielhofner) AstLinux, или приобрести Linksys WRT54GL и установить версию Asterisk, созданную для этой платформы Брайаном Капучем (Brian Capouch).

В этих проектах Asterisk представлен в базовой форме, что позволяет развертывать невероятно мощные приложения офисных АТС на очень недорогом оборудовании.

Хотя для работы с обоими проектами необходимо обладать изрядным объемом знаний и готовностью приложить большое количество усилий, они очень эффектны, чрезвычайно популярны и обеспечивают исключительное качество.

[1] Грег Бенлеин (Greg Boehnlein) однажды скомпилировал и запустил Asterisk на процессоре Pentium с частотой 133 МГц, но это был по большей мере эксперимент. Вероятность возникновения проблем с производительностью очень велика, и, чтобы сконфигурировать такую систему надлежащим образом, необходимо быть экспертом Linux. Мы не рекомендуем использовать Asterisk в системах с процессором, частота которого ниже 500 МГц (для производственной системы 2 ГГц было бы благоразумным минимумом). И все же гибкость Asterisk просто поразительна.

### Средний тип систем

Сложнее всего решать вопросы производительности именно в системах среднего типа (от 10 до 50 телефонов). Как правило, такие системы развертываются только на одном или двух серверах и, таким образом, каждая машина должна будет обрабатывать по несколько специальных задач. По мере роста нагрузки платформа все больше приближается к своим предельным значениям технических характеристик. Пользователи могут начать испытывать проблемы с качеством связи, не понимая, что это происходит не потому, что система неисправна, а просто из-за того, что достигнуты пределы ее возможностей. По мере роста нагрузки на систему проблемы будут увеличиваться, а удовлетворенность пользователей, соответственно, падать. Исключительно важно, чтобы проблемы с производительностью были выявлены и решены до того, как они будут замечены пользователями.

Отслеживание производительности в таких системах и быстрое реагирование на любые возникающие тенденции – основные условия, которые гарантируют, что платформа обеспечит качественную телефонную связь.

### Большой тип систем

Большие системы (более 120 каналов) обычно развертываются на нескольких системах и сайтах, и, таким образом, вопросы производительности можно решать путем добавления компьютеров. Очень большие системы Asterisk созданы именно так.

Построение большой системы требует наличия глубоких знаний по множеству различных дисциплин. Не будем подробно останавливаться на этом в данной книге, отметим только, что проблемы, возникающие в этом случае, будут аналогичны сложностям, которые появляются при любом использовании нескольких серверов, обрабатывающих одну распределенную задачу.

## Выбор системной платы

Просто чтобы сразу устранить любую предвзятость, мы также не будем рекомендовать конкретную системную плату в данной книге. В условиях, когда каждую неделю появляются новые системные платы, любые рекомендации станут неактуальными еще до того, как книга появится на полках книжных магазинов. Более того, системные платы подобны автомобилям: принцип один, отличия – в деталях. И поскольку Asterisk – приложение, требовательное к производительности, детали имеют большое значение.

Однако мы все-таки дадим некоторое преставление о том, какие системные платы обеспечат хорошую работу Asterisk и платы с какими характеристиками можно считать подходящими. Главное – они должны обеспечивать стабильность и высокую производительность. Вот некоторые рекомендации:

- Различные системные шины должны обеспечивать минимально возможную задержку при обработке данных. Если планируется PSTN-соединение с использованием аналогового или PRI-интерфейсов (обсуждаются в этой главе ниже), наличие в системе плат Zaptel обеспечит формирование 1000 запросов на прерывание в секунду. Наличие других устройств на шине, мешающих этому процессу, приведет к снижению качества связи. Наборы микросхем производства Intel (для процессоров Intel) и nVidia nForce (для процессоров AMD) считаются лучшими в этой области. При оценке любой системной платы проверьте ее набор микросхем, чтобы убедиться, что для него не зафиксированы случаи возникновения проблем со временем ожидания запроса на прерывание.
- При использовании в системе плат Zaptel необходимо убедиться, что BIOS[1] обеспечивает максимальный контроль над распределением прерываний. Как правило, системные платы высокого класса обеспечивают намного бóльшую гибкость при настройке BIOS; дешевые платы обычно предлагают очень ограниченные возможности управления. Однако это спорный вопрос, поскольку системные платы с включенным встроенным APIC[2] передают управление прерываниями операционной системе.
- Серверные системные платы обычно реализуют иной PCI-стандарт, нежели системные платы для рабочих станций. Различий много, но наиболее очевидное и широко известное – то, что эти две версии имеют разные напряжения. Приобретая платы, необходимо знать, какие PCI-разъемы нужны: с напряжением 3,3 или 5 В. На рис. 2.1 наглядно показано, чем отличаются разъемы 3,3 и 5 В[3]. На большинстве серверных системных плат есть оба типа разъемов, но платы для рабочих станций обычно имеют только разъем 5 В.

Есть свидетельство тому, что объединение двух совершенно независимых однопроцессорных систем может обеспечить намного больше преимуществ, чем использование двух процессоров в одном компьютере. В этом случае не только удваивается мощность процессора, но также достигается намного лучший уровень избыточного резервирования по цене, равной стоимости компьютера с одним системным блоком и двумя процессорами. Однако нельзя забывать, что спроектировать решение Asterisk с двумя серверами намного сложнее, чем с одним компьютером.

---

1 Basic Input-Output System, BIOS – базовая система ввода/вывода. – *Примеч. науч. ред.*

2 Advanced Programmable Interrupt Controller, APIC – усовершенствованный программируемый контроллер прерываний. – *Примеч. науч. ред.*

3 С появлением PCI-X и PCI-Express становится все сложнее правильно выбрать системную плату с соответствующими типами разъемов. При покупке необходимо удостовериться, что тип и количество разъемов для плат соответствуют имеющемуся оборудованию. Большинство компаний, выпускающих платы для Asterisk, предлагают и PCI, и PCI-Express, а уж вам решать, какой вариант подойдет для выбранного сочетания системной платы и системного блока.

- Рассмотрите вариант использования нескольких процессоров или процессоров с несколькими ядрами. Это улучшит возможность системы обрабатывать несколько задач, а для Asterisk предоставит особые преимущества при выполнении операций с плавающей точкой.
- Если требуется модем, лучше установить внешнее устройство, подключаемое через последовательный порт. Если должен использоваться внутренний модем, необходимо убедиться, что это не так называемый Win-модем[1], это должно быть абсолютно автономное устройство (заметьте, что такое устройство очень сложно или практически невозможно найти).
- Следует учесть, что при использовании встроенных сетевых устройств в случае их выхода из строя придется заменить всю системную плату. С другой стороны, если устанавливается внешняя сетевая интерфейсная плата (Network Interface Card, NIC), вероятность поломки возрастает из-за присутствия большого количества механических соединений. Также может быть целесообразным использование разных сетевых плат для телефонов и пользователей (внутренней сети) и провайдеров VoIP и внешних сайтов (внешней сети). Сетевые адаптеры стоят недорого; рекомендуем всегда иметь под рукой по крайней мере пару.
- Стабильность и качество системы Asterisk будет зависеть от компонентов, выбранных для ее архитектуры. Asterisk – хищник, его надо очень хорошо «кормить». Но, как практически во всем, высокая цена не всегда является синонимом качества. Вы должны будете стать знатоком компьютерных комплектующих.

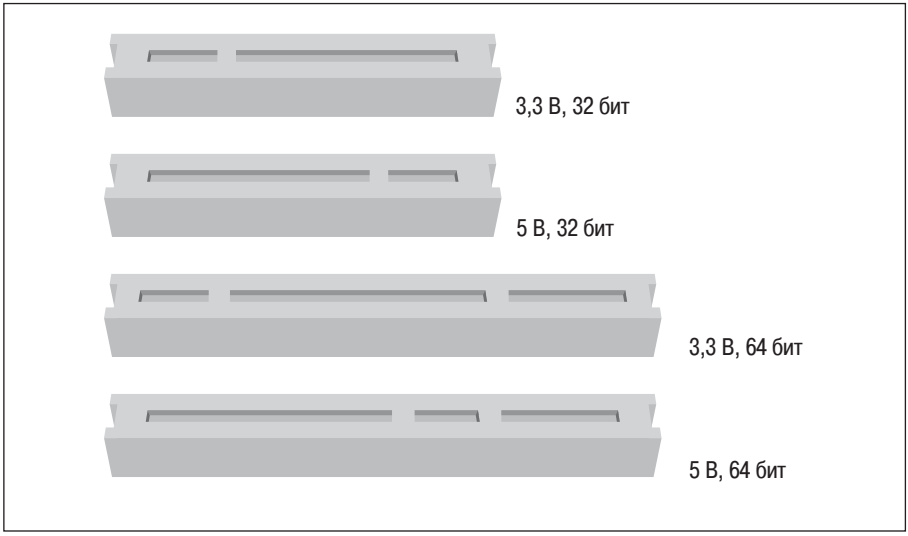



***Рис. 2.1.*** *Внешний вид PCI-разъемов*

[1] Также такие модемы иногда называют soft-модемами. – *Примеч. науч. ред.*

Обсудив все это, мы должны вернуться к исходной точке: Asterisk может и будет замечательно устанавливаться практически на любую систему, работающую под управлением Linux. Лабораторные системы, использовавшиеся при написании этой книги, например, включали все, от Linksys WRT до «трактора» dual-Xeon[1]. У нас не возникало никаких проблем с производительностью или стабильностью при установлении до пяти одновременных соединений. В целях обучения Asterisk можно устанавливать на любую имеющуюся в распоряжении систему. Однако, когда вы будете готовы создавать системы для эксплуатации, необходимо понимать последствия принимаемых решений об использовании того или иного оборудования.

## Требования к блоку питания

Блоку питания (и вопросу электропитания) ПК обычно уделяется недостаточно внимания. Для телекоммуникационных систем[2] эти компоненты могут играть важную роль в формировании хорошего впечатления у пользователя.

### Блоки питания для компьютеров

Выбранный для системы блок питания будет играть жизненно важную роль в стабильности всей платформы. Asterisk не является особенно энергоемким приложением, но все, что связано с мультимедийными системами (будь то телефония, профессиональная аудио-, видеоаппаратура и т. п.), обычно чрезвычайно чувствительно к *качеству* электропитания.

Этот часто упускаемый из виду компонент может превратить высококачественную систему в груду хлама. Справедливо и то, что с первоклассным блоком питания дешевый ПК может стать настоящим чемпионом.

Получаемая мощность должна не только удовлетворять потребности системы в энергии, необходимой для выполнения ее задач, но также обеспечивать стабильные и четкие сигнальные линии для всех уровней напряжений, ожидаемых системой.

Не пожалейте денег и приобретите высококлассный блок питания (геймеры отличаются особенно трепетным отношением к таким вещам, поэтому их выбор на рынке очень богат).

### Блоки питания с резервированием

В средах операторского класса или бесперебойной работы принято развертывать серверы, использующие блок питания с резервированием. Фактически это два совершенно независимых блока питания, каждый из которых может полностью обеспечить требования по питанию системы.

---

[1] Ну конечно же, это был *не настоящий* трактор, но шума от него было столько же. Кто-нибудь знает, где достать бесшумные вентиляторы для процессоров Xeon? Очень уж шумно в лаборатории.

[2] А также любой системы, которая должна обрабатывать аудиоданные.

Опыт показывает, что для правильного резервирования такие блоки питания должны быть подключены к совершенно независимым источникам бесперебойного питания (Uninterruptible Power Supplies, UPSes), которые, в свою очередь, питаются от разных электрических сетей. В ответственных организациях (таких, например, как больницы) резервируются даже основные подводы электропитания зданий и для генерирования электричества во время длительных перебоев с электроэнергией (подобных, например, случившемуся на северо-востоке США 15 августа 2003 года) используются дизельные генераторы.

# Окружение

Окружение системы образуют все факторы, которые сами по себе фактически не являются частью сервера, но тем не менее играют важную роль в формировании ожидаемых от системы надежности и качества. Электропитание, температура и влажность помещения, источники помех и безопасность – все эти факторы должны быть учтены.

## Обеспечение требуемого качества электроэнергии и источники бесперебойного питания

При выборе источников питания для своей системы необходимо учесть не только потребляемую мощность, но также то, как эта энергия будет обеспечиваться.

Электроэнергия – это не просто напряжение в розетке на стене, и производственную систему никогда нельзя подключать к первому попавшемуся источнику питания[1]. Продумав схему электропитания своей системы, можно создать намного более стабильное энергоокружение, что обеспечит наибольшую стабильность системы.

Одно из преимуществ соответствующей стандартам энергии хорошего качества – снижение тепловыделения, что означает меньшую нагрузку на компоненты и увеличение их срока службы.

Правильно заземленная, отвечающая стандартам электрическая сеть и высококачественный блок питания гарантируют четкий опорный сигнал «логическая земля» (то есть 0 В[2]) для системы и минимальный

---

[1] Ну ладно, систему *можно* подключить к чему угодно, и она даже, вероятно, будет работать, но когда в ней возникнут непонятные проблемы со стабильностью, перечитайте этот раздел, пожалуйста. Договорились?

[2] В электронных устройствах двоичный нуль (0) обычно обозначает сигнал 0 В, тогда как двоичная единица (1) может быть представлена разными напряжениями (обычно в диапазоне от 2,5 до 5 В). Напряжение «земли», за которое система будет принимать 0 В, часто называют *логическим нулем*. Плохо заземленная система может иметь такой электрический потенциал на логическом нуле, что устройство сможет принять двоичный нуль за двоичную единицу. Это может серьезно навредить способности системы обрабатывать команды.

электрический шум[1] на системной плате. Для данного типа оборудования существуют лучшие практики, принятые как отраслевой стандарт, которыми нельзя пренебрегать. Относительно простой способ обеспечить это – использовать UPS с поддержанием требуемого качества электроэнергии[2].

### UPS с поддержанием требуемого качества электроэнергии

UPS широко используются как аккумуляторы для снабжения системы питанием в случае аварии, хотя возможность высококачественных UPS обеспечивать электроэнергию в соответствии с техническими требованиями иногда не принимается во внимание.

Обеспечение требуемого качества электроэнергии создаст необходимый уровень защиты от электрической сети общего пользования за счет генерации соответствующего стандартам напряжения посредством разделительного трансформатора. Качественный стабилизатор напряжения в UPS устранит бóльшую часть электрического шума, поступающего из питающей электросети, и гарантирует снабжение системы энергией в течение длительного периода.

К сожалению, не все UPS одинаковы; многие более дешевые модели не обеспечивают качественной стабилизации напряжения. Что хуже всего, производители этих устройств часто обещают все виды защиты от скачков, бросков, повышения напряжения и импульсных помех. Хотя такие устройства могут защитить систему от воспламенения во время грозы, они не приведут напряжение в норму на входе в систему и, таким образом, не сделают ничего для обеспечения стабильности.

Убедитесь, что у вас *UPS с поддержанием требуемого качества электроэнергии*. Если это нигде прямо не указано, значит, он таковым не является.

## Заземление

*Напряжение* определяется как разность потенциалов между двумя точками. Обычно считается, что напряжение «земли» (а это, по сути, не что иное как токопровод к земле) равно 0 В. Но если не определить эти 0 В *относительно* чего-то, мы рискуем делать предположения, не соответствующие действительности. Часто между двумя точками заземления существует некоторая разность потенциалов. Этого напряжения может быть достаточно для возникновения логических ошибок или даже повреждения системы, в которой имеется несколько контуров заземления.

---

[1] Колебания напряжения в электрической сети. – *Примеч. науч. ред.*

[2] Общепринятым заблуждением является мнение о том, что UPS обеспечивают соответствующую стандартам энергию хорошего качества. Это абсолютно не соответствует действительности.

Один из авторов данной книги вспоминает, как сжег звуковую карту, пытаясь подключить ее к стереосистеме друга. Даже несмотря на то что компьютер и стереосистема находились в одной комнате, между заземлителями двух электрических розеток, в которые подключались устройства, было замерена разность потенциалов в 6 В! Провод между стереосистемой и ПК (посредством звуковой карты) обеспечил свободную передачу этого напряжения, что сожгло звуковую карту, которая не была рассчитана на такой большой ток в сигнальном проводе. Подключение ПК и стереосистемы в одну розетку решило проблему.

Согласно правилам устройства электроустановок заземление – это, главным образом, средство обеспечения безопасности человека. В компьютере понятие «земля» используется для обозначения логического опорного сигнала 0 В. Электрическая система, обеспечивающая должную безопасность, не всегда будет обеспечивать соответствующее логическое опорное напряжение. Надо отметить, что задачи безопасности иногда идут вразрез с задачами по обеспечению качества напряжения. Естественно, если необходимо сделать выбор, безопасность должна быть на первом месте.

Поскольку разница между двоичным нулем и двоичной единицей представлена в компьютерах разностью напряжений, которая иногда меньше, чем 3 В, в условиях нестабильного напряжения, обусловленных плохим заземлением или электрическим шумом, вполне вероятно периодическое возникновение всевозможных проблем в системе. Некоторые исследователи вопросов напряжения и заземления утверждают, что 80% необъяснимых сбоев компьютеров происходит из-за неудовлетворительного качества электропитания. А большинство из нас ругают Майкрософт.

Современные импульсные источники питания несколько ушли от проблем с качеством электроэнергии, но любая высокопроизводительная система всегда выиграет от хорошо спроектированной системы электропитания. В больших ЭВМ, специализированных офисных АТС и на других дорогих вычислительных платформах вопросы заземления системы никогда не пускаются на самотек. Электроника и стойки этих систем всегда снабжены выделенным заземлением, которое не связано с защитным заземлением, поставляемым с подводом электропитания.

Независимо от того, сколько предполагается инвестировать в заземление, при определении системы электропитания для любой офисной АТС убедитесь, что в данную электрическую сеть подключена только ваша система (это обсуждается в следующем разделе) и что имеется от-

дельный изолированный заземляющий провод. Это может быть дорого, но значительно повысит качество энергоокружения системы[1].

Также жизненно важно, чтобы абсолютно все периферийные устройства, подключаемые к системе, были включены в одну розетку (или, более конкретно, на одно заземление). Это сократит вероятность возникновения контуров заземления, которые могут вызвать все что угодно, начиная от шумов и заканчивая повреждением и выходом из строя оборудования.

## Электрические сети

Если вам доводилось видеть мерцание освещения при подключении какого-либо электроприбора, вы были свидетелем воздействия мощного устройства на электросеть. Если бы вы взглянули на эффект от подключения множества подобных устройств, каждое из которых вносит дополнительную нагрузку на сеть, вы бы увидели, что система получает все что угодно, но не идеальную гармоническую волну с частотой 50 или 60 Гц. Гармонический шум – крайне распространенное явление в электрических сетях, и он может нанести непоправимый вред чувствительной электронной аппаратуре. Для офисных АТС эти проблемы могут проявляться как проблемы со звуком, логические ошибки и нестабильность системы.

В идеале сервер никогда не должен подключаться к электросети совместно с другими устройствами. В сети должна быть одна розетка, и в нее должна быть подключена только телефонная система (и ее периферийные устройства). Провода (включая заземление) должны проходить все вместе сразу за электрораспределительным щитом. Заземлитель должен быть изолирован и отделен от остальных проводов. Известно слишком много историй о том, как ксероксы, кондиционеры и пылесосы выводили из строя чувствительную электронику при пренебрежении этим правилом.

Региональные правила устройства электроустановок являются более приоритетными, чем все высказанные здесь рекомендации. В случае возникновения каких-либо сомнений необходимо обратиться за консультацией к региональному специалисту по качеству электроэнергии по поводу того, как обеспечить соответствие системы существующим правилам. Помните, правила устройства электроустановок определены исходя из условия, что безопасность человека выше безопасности оборудования.

[1] Для любительской системы, возможно, это излишнее требование, но если Asterisk планируется использовать для каких-нибудь важных целей, убедитесь, по крайней мере, что дали системе шанс на выживание; не включайте с ней в одну сеть кондиционеры воздуха, светокопировальные аппараты, лазерные принтеры, электродвигатели и т. п. Нагрузка, создаваемая подобными устройствами на сеть, сократит срок службы системы.

## Аппаратная комната

Внешние условия эксплуатации могут иметь очень негативное влияние на устройства, но по-прежнему очень часто можно увидеть, что критически важные системы эксплуатируются с недостаточным учетом внешних факторов или совершенно без него. Сразу после установки все работает хорошо, но не проходит и полугода, как компоненты начинают выходить из строя. Если пообщаться с людьми, имеющими опыт обслуживания серверов и систем, становится очевидным, что внимательное отношение к факторам окружающей среды может играть важную роль в стабильности и надежности систем.

### Влажность

Попросту говоря, влажность – это вода, присутствующая в воздухе. Вода губительна для электроники по двум главным причинам: 1) она является катализатором коррозии, и 2) вода обладает достаточной проводимостью и может быть причиной коротких замыканий. Электронное оборудование нельзя устанавливать в зонах с повышенной влажностью без обеспечения способов удаления влаги.

### Температура

Повышенная температура – враг электроники. Чем прохладнее место, где находится система, тем надежнее и дольше она работает. Если в помещении, где эксплуатируется система, невозможно обеспечить адекватное охлаждение, как минимум, необходимо гарантировать непрерывный приток чистого холодного воздуха к системе. Также должен быть обеспечен постоянный температурный режим. Перепады температуры могут привести к конденсации влажности и другим опасным изменениям.

### Пыль

В компьютерном фольклоре жива старая поговорка, что пыль внутри компьютера – к удаче. Давайте рассмотрим, к чему приводит пыль в реальности:

- Значительные накопления пыли могут затруднять циркуляцию воздуха в системе, приводя к повышению уровня температуры.
- В пыли могут содержаться частицы металлов, которые в достаточных количествах будут способствовать деградации сигнала или приводить к коротким замыканиям на печатных платах.

Важные серверы необходимо размещать в помещениях с фильтрацией воздуха и выполнять регулярную очистку оборудования.

### Безопасность

Безопасность серверов обычно включает защиту от вторжений со стороны сети, но условия эксплуатации также играют роль в безопасности системы. Телефонное оборудование должно быть «под замком», доступ к нему разрешается только обслуживающему персоналу.

# Оборудование для телефонии

Если предполагается соединение Asterisk с любым традиционным телекоммуникационным оборудованием, потребуются соответствующие аппаратные средства. Ответ на вопрос о том, какое оборудование понадобится, будет зависеть от того, чего конкретно необходимо достичь.

## Подключение к PSTN

Asterisk позволяет эффективно связывать коммутируемые телекоммуникационные сети[1] с сетями передачи данных с коммутацией пакетов[2]. Открытая архитектура (и открытый исходный код) Asterisk позволяет соединять любое соответствующее стандартам интерфейсное оборудование. Выбор интерфейсных плат для телефонии с открытым исходным кодом в настоящее время ограничен, но, поскольку интерес к Asterisk растет, эта ситуация быстро изменится[3]. На данный момент одним из наиболее популярных и рентабельных способов подключения к PSTN является использование интерфейсных плат, разработанных в рамках проекта Zapata Telephony Project (*http:// www.zapatatelephony.org*).

### Аналоговые интерфейсные платы

Интерфейс PSTN, скорее всего (если не требуется обеспечение многоканальной линии или нет денег на то, чтобы каждый месяц менять телекоммуникационное оборудование), будет состоять из одной или более аналоговых схем, для каждой из которых потребуется порт Foreign eXchange Office (FXO).

Digium, компания, спонсирующая разработку Asterisk, выпускает аналоговые интерфейсные платы для Asterisk. На веб-сайте компании можно найти обширный модельный ряд аналоговых карт, включая проверенную временем TDM400P, новейшую TDM800P и плату с высокой плотностью размещения проводников TDM2400P. Например, TDM800P – восьмипортовая базовая плата с возможностью установки максимум двух четырехпортовых модулей FXO или FXS[4]. Можно при-

---

[1] Часто их называют *TDM-сетями* из-за технологии Time Division Multiplexing (мультиплексирование с разделением по времени), применяемой для переноса трафика по PSTN.

[2] Обычно их называют VoIP-сетями, хотя передача голоса по IP-протоколу – не единственный способ передачи речи по сетям пакетной коммутации (передача голоса по сети Frame Relay была очень популярна в конце 1990-х годов).

[3] Эволюция недорогого телефонного оборудования для широкого потребления лишь немного отстает от революции в программном обеспечении для телефонии. Каждую неделю возникают новые компании и поставляют на рынок новые недорогие и отвечающие стандартам устройства.

[4] FXS и FXO являются альтернативным оборудованием для аналоговой схемы. Какой из них потребуется, будет определяться тем, к чему выполняется подключение. Этот вопрос более подробно обсуждается в главе 7.

обрести TDM800P с уже установленными данными модулями, а также с модулем для эхоподавления. Более подробную информацию об этих платах можно найти на сайте компании Digium (*http://www.digium.com*).

Производством совместимых с Asterisk аналоговых плат также занимаются следующие компании:

- Rhino (*http://www.channelbanks.com*).
- Sangoma (*http://www.sangoma.com*).
- Voicetronix (*http://www.voicetronix.com*).
- Pika Technologies (*http://www.pikatechnologies.com*).

Все это компании с хорошей репутацией, выпускающие превосходные продукты.

## Цифровые интерфейсные платы

Если требуется более 10 телефонных линий или обеспечение возможности подключения к цифровым линиям, используются платы T1 или E1[1]. Однако не стоит забывать, что ежемесячные расходы на обслуживание цифровых PSTN-линий варьируются в широких пределах. В некоторых регионах окупаемость будет обеспечена всего пятью линиями; в других эта технология может вообще не быть экономически эффективной. Чем выше конкуренция в регионе, тем больше шансов найти более выгодное предложение. Взвесьте все возможные варианты.

В ходе проекта Zapata Telephony Project сначала была создана плата T1, Tormenta, то есть прототип наиболее совместимых с Asterisk плат T1. Первые платы Tormenta сейчас считаются устаревшими, но они до сих пор работают с Asterisk.

Digium выпускает несколько разных интерфейсных плат для цифровых линий. Эти платы практически идентичны; основное отличие в предоставляемых интерфейсах, T1 или E1, и количестве обеспечиваемых каналов. Компания Digium дольше всех выпускает платы Zaptel для Linux, поскольку принимала активное участие в разработке Zaptel под Linux и с годами стала движущей силой разработки Zaptel.

Sangoma, которая выпускает платы WAN с открытым исходным кодом в течение многих лет, добавила поддержку Asterisk для своих плат T1/E1 несколько лет назад[2]. Сейчас Rhino выпускает платы T1 для Asterisk. Также существует много других компаний, предлагающих цифровые интерфейсные платы для Asterisk.

---

[1] T1 и E1 – это потоки, используемые для цифровых телефонных линий. Они обсуждаются подробнее в главе 7.

[2] Следует заметить, что плата Frame Relay производства Sangoma играла некоторую роль в первоначальной разработке Asterisk (см. *http://linuxdevices.com/articles/AT8678310302.html*); Sangoma имеет долгую историю поддержки интерфейсов WAN с открытым исходным кодом в Linux.

## Банки каналов

*Банк каналов* – это, грубо говоря, устройство, позволяющее разделить цифровую линию на несколько аналоговых линий (и наоборот). Выражаясь точнее, банк каналов обеспечивает возможность объединять аналоговые телефоны и линии в систему через линию T1. На рис. 2.2 показан банк каналов в составе типовой офисной телефонной системы.

Несмотря на дороговизну, многие считают, что использование банка каналов – это единственно правильный способ объединения аналоговых линий и устройств с Asterisk. Так это или нет, зависит от многих факторов, но, если вы можете себе это позволить, лучше не экономьте на банке каналов[1]. Часто уже бывшие в употреблении банки каналов можно найти на аукционе eBay. Ищите модули компаний Adtran и Carrier Access Corp. (Rhino делает замечательные банки каналов, и они очень привлекательны по цене, но на eBay их трудно найти.) Не следует забывать, что для подключения банка каналов к Asterisk понадобится плата T1.

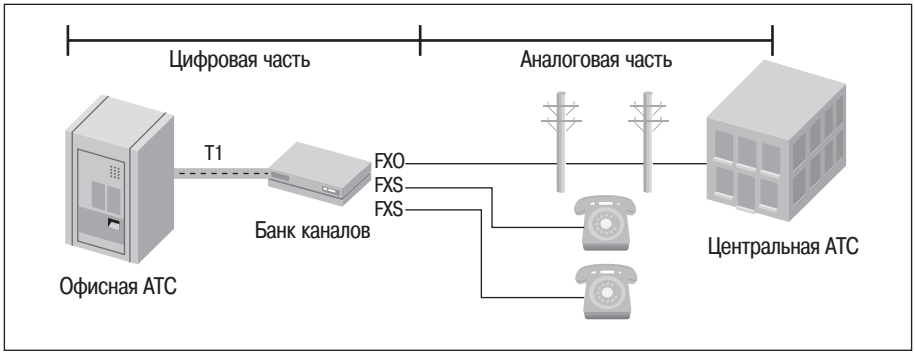



***Рис. 2.2.*** *Один из способов подключения банка каналов*

## Другие типы интерфейсов PSTN

Существует множество шлюзов VoIP, которые можно конфигурировать для обеспечения доступа к линиям PSTN. Вообще говоря, больше всего они пригодятся в небольших системах (одна или две линии). Процесс конфигурирования также может вызывать сложности, потому что управление взаимодействием между различными сетями и устройствами требует твердого понимания и телефонии, и основ VoIP. По этой причине упомянутые устройства не будут обсуждаться в данной книге подробно, однако они заслуживают внимания. Популярностью пользу-

[1] Мы используем банки каналов для моделирования центральной АТС. Один 24-портовый банк каналов на выходе из системы Asterisk может обеспечить до 24 аналоговых линий – это идеально для аудитории или лаборатории.

ются модули, выпущенные компаниями Sipura, Grandstream, Digium и многими другими.

Другой способ подключения к PSTN – посредством линий ISDN[1] Basic Rate Interface (BRI). BRI[2] – это цифровой телекоммуникационный стандарт, определяющий двухканальную линию с пропускной способностью до 144 Кбит/с. Он очень редко используется в Серверной Америке, но крайне популярен в Европе. Из-за большого многообразия способов реализации данной технологии и отсутствия оборудования для тестирования в данной книге мы не будем останавливаться на BRI слишком подробно. Но, пожалуйста, обратите внимание, что BRI очень популярен в Европе и поэтому Digium выпустила плату B410P.

## Соединение исключительно с телефонной сетью на базе коммутации пакетов

Если нет необходимости подключения к PSTN, для Asterisk не требуется никакого другого оборудования, кроме сервера с сетевой интерфейсной платой.

Однако, если предполагается предоставление возможности воспроизведения музыки при ожидании[3] или конференц-связи и нет физического источника временных интервалов, понадобится модуль ядра Linux *ztdummy*. ztdummy – это генератор тактовых синхронизирующих импульсов, разработанный как источник временных интервалов для систем, не имеющих аппаратного таймера. Его можно рассматривать как своего рода метроном, благодаря которому система может правильно синхронизировать множество аудиопотоков при их смешении.

## Эхоподавление

Одной из проблем, возникающих при использовании аналоговых интерфейсов в системе VoIP – эхо. *Эхо* – это возвращение сказанного говорящему через небольшой промежуток времени. Эхо возникает на противоположном конце линии связи, но слышит его говорящий на этом конце. Малоизвестный факт, что эхо было бы огромной проблемой в PSTN, если бы поставщики услуг связи не применяли сложные (и дорогие) стратегии для его устранения. Мы поговорим об эхе немного позже, но, что касается аппаратных средств, советуем подумать о добавлении эхоподавляющего оборудования на все платы, приобретаемые для использования в качестве интерфейса PSTN. Asterisk мо-

---

[1] Цифровая сеть с интеграцией служб. – *Примеч. науч. ред.*

[2] Интерфейс, обеспечивающий базовую скорость передачи данных. – *Примеч. науч. ред.*

[3] С технической точки зрения для воспроизведения музыки при ожидании источник временных интервалов не нужен, но обычно эта функция выполняется лучше при наличии такового.

жет кое-что делать с эхом на программном уровне, но этого отнюдь не достаточно для решения проблемы. Эхоподавление на программном уровне очень сильно загружает процессор; аппаратные эхокомпенсаторы, встроенные в PSTN-плату, снимают эту ношу с ЦП.

Аппаратное эхоподавление может увеличить стоимость оборудования на несколько сотен долларов, но, если требуется получить качественную систему, лучше вложить немного больше средств сейчас, чем иметь неприятности потом. Эхо – очень неприятная проблема, и пользователи просто возненавидят систему, в которой такая проблема присутствует.

На момент написания данной книги на рынке появилось несколько программных эхокомпенсаторов. У нас не было возможности протестировать ни один из них, но известно, что в них используются те же алгоритмы, что и в аппаратных эхокомпенсаторах. Те, кто недавно приобрел аналоговую плату Digium, могут позвонить в отдел продаж Digium и получить код ключа, с помощью которого вы сможете воспользоваться их самым последним программным эхокомпенсатором в своей системе[1]. Есть и иные варианты ПО для других типов плат, но необходимо выяснить, нужна ли лицензия на их использование[2]. Помните, что применение эхокомпенсаторов приводит к потерям производительности. Они существенно нагружают процессор, и это следует учитывать при проектировании системы, использующей данные технологии.

## Типы телефонов

Эта книга называется «Asterisk: будущее телефонии», и с нашей стороны было бы небрежностью не упомянуть об устройствах, с которыми в конечном счете будет взаимосвязана данная технология: телефоны!

Всем известно, что такое телефон, но останется ли он таким же через пять лет? Революционные преобразования, которым способствует Asterisk, включают и эволюцию телефона: от простого устройства аудиосвязи до мультимедийного терминала связи, предоставляющего всевозможные функции, которые пока что сложно даже представить.

В качестве введения в эту увлекательную область кратко рассмотрим различные виды устройств, называемые в настоящее время «телефонами» (все они без труда могут быть интегрированы с Asterisk). Также немного пофантазируем о том, во что могут развиться эти устройства

---

[1] Это ПО не входит в обычный пакет для скачивания Asterisk, потому что Digium приходится платить за его лицензию отдельно. Тем не менее оно поставлялось со всеми платами Digium до введения действующего закона в силу, поэтому доступно бесплатно всем, у кого есть аналоговая плата Digium, находящаяся на гарантийном обслуживании. Если вы используете аналоговую плату другого производителя, на веб-сайте Digium можно купить ключ для этого программного эхокомпенсатора.

[2] Sangoma тоже предлагает бесплатный программный эхокомпенсатор на своих аналоговых платах (до шести каналов).

в будущем (в устройства, которые также будут запросто интегрироваться с Asterisk).

## Физические телефоны

Любое физическое устройство, основным назначением которого является замыкание по требованию линии аудиосвязи между двумя точками, можно классифицировать как физический телефон. Как минимум, такое устройство имеет телефонную трубку и номеронабиратель. Также могут присутствовать функциональные клавиши, дисплей индикатора и различные аудиоинтерфейсы.

В данном разделе дается краткое описание различных пользовательских (или терминальных) устройств, которые могут быть включены в систему Asterisk. Более детально техническое оснащение аналоговой и цифровой телефонии рассматривается в главе 7.

### Аналоговые телефоны

Аналоговые телефоны существуют с момента появления телефонии. Еще примерно 20 лет назад все телефоны были аналоговыми. В разных странах производятся немного разные аналоговые телефоны, но принцип их работы везде одинаковый.

Непрерывное соединение называют *каналом (или линией) связи*, для их установления в телефонной сети использовались электромеханические коммутаторы – отсюда и название: *сеть с коммутацией каналов (или линий)*.

Когда человек начинает говорить, голосовые связки, язык, зубы и губы формируют множество сложных звуков. Назначение телефона – улавливать эти звуки и преобразовывать их в формат, пригодный для передачи по проводам. В аналоговой телефонии передаваемый сигнал является *аналогом* звуковых волн, создаваемых говорящим объектом. Если бы можно было увидеть звуковые волны, поступающие от рта к микрофону, они были бы пропорциональны электрическому сигналу, который можно замерить в проводе.

Аналоговые телефоны – это лишь самая общедоступная разновидность телефона. В следующие несколько лет ожидаются разительные перемены.

### Специализированные цифровые телефоны

С развитием цифровых систем коммутации в 1980-х и 1990-х годах телекоммуникационные компании разработали цифровые офисную АТС (Private Branch eXchanges, PBXes) и малую АТС (Key Telephone Systems, KTSes). Разработанные для них специализированные телефоны были полностью зависимы от систем, к которым подключались, и не могли использоваться в других системах. Совместимыми не были

даже телефоны одного производителя (например, аппарат Nortel Norstar не работает с офисной АТС Nortel Meridian 1). Из-за такой узкой специализированности и несовместимости цифровые телефоны не имеют будущего. В начинающуюся ныне эпоху стандартизованной связи они быстро окажутся на свалке истории.

Аппарат цифрового телефона обычно функционально идентичен аналоговому телефонному аппарату, и часто они совместимы друг с другом. Чем цифровой телефон отличается от аналогового, так это тем, что происходит у него внутри: аналоговый сигнал дискретизируется и преобразуется в цифровой, то есть в численное представление аналоговой волны. Подробное обсуждение цифровых сигналов отложим до главы 7. Пока достаточно лишь сказать следующее: основное преимущество цифрового сигнала в том, что он может передаваться на неограниченные расстояния без потери качества.

Шансы, что когда-нибудь будет создан специализированный цифровой телефон, полностью совместимый с Asterisk, малы, но компании, такие как Citel (*http://www.citel.com*)[1], разработали шлюзы, преобразующие специализированные сигналы в сигналы, соответствующие протоколу Session Initiation Protocol (SIP)[2].

## ISDN-телефоны

До появления VoIP ближе всего к стандартизованному цифровому телефону был терминал ISDN BRI. Разработанный в начале 1980-х годов, ISDN должен был совершить переворот в телекоммуникациях и сделать именно то, что обещает наконец довести до конца VoIP сегодня.

Существует два типа ISDN: *Primary Rate Interface* (PRI) и *Basic Rate Interface* (BRI). PRI обычно используется для обеспечения соединений между офисными АТС и PSTN и широко распространен во всем мире. BRI не используется в Северной Америке, зато популярен в Европе.

Хотя ISDN широко используется телефонными компаниями, многие считают этот стандарт неудачным, поскольку он, в целом, не оправдал ожиданий. Высокая стоимость реализации, необходимость периодически делать повторные капиталовложения и отсутствие сотрудничества между основными игроками на рынке – все это создает больше проблем, чем решает данная система.

---

[1] Citel выпустили фантастический продукт, единственным недостатком которого является его дороговизна. Если имеются специализированные телефоны старой офисной АТС, применяя технологию Citel, их можно использовать с системой Asterisk. Но сравните, сколько придется потратить на приобретение этих модулей, по одному на порт, со стоимостью замены старых аппаратов просто телефонами VoIP.

[2] В настоящее время SIP – самый известный и популярный протокол для VoIP. Он будет обсуждаться в главе 8.

BRI предназначался для обслуживания терминалов и меньших узлов связи (контур связи BRI обеспечивает два цифровых канала). Было разработано множество BRI-устройств, однако BRI был преимущественно отвергнут в пользу более быстрых и дешевых технологий, таких как ADSL[1], кабельные модемы и VoIP.

BRI по-прежнему очень широко используется как оборудование для видеоконференц-связи, поскольку обеспечивает линию с фиксированной полосой пропускания. Также для BRI не характерны проблемы с качеством, какие могут возникать при VoIP-соединении, поскольку это интерфейс с коммутацией каналов.

BRI до сих пор иногда используется вместо аналоговых линий, чтобы обеспечить соединение с офисной АТС. Хороша эта идея или плоха – зависит преимущественно от цен на эту услугу, устанавливаемых локальной телефонной компанией, и от того, какие возможности она желает предоставлять[2].

### IP-телефоны

IP-телефоны – вестники наиболее захватывающего изменения в телекоммуникационной отрасли. Уже сейчас IP-телефоны, отвечающие стандартам, можно найти в розничной торговле. Богатство возможностей, предлагаемых этими устройствами, обусловит шквал любопытнейших применений, начиная от видеотелефонов до устройств для вещания с высоким качеством, беспроводных мобильных решений, специализированных телефонных аппаратов, предназначенных для конкретных отраслей, и гибких мультимедийных систем «все в одном».

Революция, инициируемая IP-телефонами, отнюдь не заключается в появлении нового типа сети, к которой можно будет подключить свой телефон; речь идет исключительно об обеспечении мощной возможности общаться именно так, как вам хочется.

По первым моделям IP-телефонов, появившимся несколько лет назад, нельзя судить о будущих возможностях этих замечательных устройств. Это просто трамплин, знакомая обертка, в которой следует преподнести новый фантастический образ мышления.

Будущее предвещает намного большее.

## Программные телефоны

*Программный телефон* – это приложение, которое обеспечивает функциональность телефона устройству, не являющемуся телефоном, такому как ПК или персональный цифровой секретарь. Итак, на что это

---

[1] Asymmetric Digital Subscriber Line – асимметричная цифровая абонентская линия. – *Прим. науч. ред.*

[2] В США, не располагая большим количеством денег и терпения, лучше отказаться от этой идеи.

похоже? На первый взгляд, казалось бы, простой вопрос, но он на самом деле влечет за собой множество других. Вероятно, у программного телефона должен быть какой-то номеронабиратель и интерфейс, напоминающий пользователям телефон. Но так ли это будет?

Можно ожидать, что значение термина *«программный телефон»* будет быстро меняться по мере того, как наше представление о телефоне будет претерпевать коренные изменения[1]. В качестве примера этой эволюции рассмотрим следующее: можно ли считать популярные программы для мгновенной передачи сообщений, такие как Instant Messenger, программными телефонами? IM предоставляет возможность начинать и принимать стандартизованные VoIP-соединения. Разве это не дает право называть его программным телефоном? Чтобы ответить на этот вопрос, надо уметь заглядывать в будущее, чему мы еще не научились. Достаточно сказать следующее: хотя на данный момент ожидается, что программные телефоны будут выглядеть как традиционные телефоны, в самом ближайшем будущем эта концепция, скорее всего, изменится.

По мере изменения стандартов и ухода от традиционного телефона в направлении к культуре мультимедийной связи грань между программными и физическими телефонами будет стираться. Например, в качестве телефона можно будет приобрести терминал связи, а для получения необходимой функциональности установить на него приложение программного телефона.

Теперь, после того как мы все так запутали, лучшее, что можно сделать, – это дать определение тому, что подразумевается под термином *«программный телефон»* в данной книге, понимая при этом, что он может существенно измениться в ближайшие несколько лет. Для наших целей мы определяем программный телефон как любое устройство, выполняемое на персональном компьютере, имеющее вид и создающее впечатление телефона и обеспечивающее его основную функцию – возможность устанавливать одновременную двустороннюю аудиосвязь (что ранее называлось «телефонными звонками»)[2] через адресацию E.164[3].

## Телефонные адаптеры

*Телефонный адаптер* (обычно называемый ATA, или аналоговым терминальным адаптером) можно описать как устройство для конечного потребителя, которое обеспечивает объединение линий связи, исполь-

[1] Вы ведь имеете представление о Skype?

[2] Думаете, вы знаете, что такое звонок по телефону? Мы тоже так думали. Давайте просто подождем несколько лет, хорошо?

[3] E.164 – это стандарт МСЭ, определяющий порядок присвоения телефонных номеров. Если вы использовали телефон, вы использовали адресацию E.164.

зующих разные протоколы. Чаще всего эти устройства используются для преобразования цифрового сигнала (IP или специализированного) в аналоговый, с которым могут работать стандартные телефоны или факсы.

Такие адаптеры можно было бы называть шлюзами, потому что это – их функция. Однако популярный термин *«телефонный шлюз»*, вероятно, лучше всего описал бы многопортовый телефонный адаптер, как правило, выполняющий более сложные функции маршрутизации.

Телефонные адаптеры будут употребляться до тех пор, пока существует необходимость соединять несовместимые стандарты и старые устройства с новыми сетями. Со временем необходимость в этих устройствах отпадет, как это случилось с модемами, которые постепенно исчезают ввиду ненужности.

## Терминалы связи

*«Терминал связи»* – это старый термин, исчезнувший на пару десятков лет и воспроизведенный здесь, пожалуй, лишь по той причине, что его надо обсудить, прежде чем он в конечном счете исчезнет вновь или станет повсеместно распространенным.

Сначала немного истории. Когда были выпущены первые цифровые системы офисных АТС, производители этих машин поняли, что не могут называть их конечные точки телефонами – специализированная природа обусловливала невозможность их соединения с PSTN. Поэтому их назвали *терминалами* или *станциями*. Конечно, пользователи этого не приняли. Эти системы выглядели как телефон и работали как телефон, то есть *были* телефонами. По-прежнему изредка можно встретить термин «терминал» в применении к аппаратам офисной АТС, но преимущественно их называют телефонами.

Обновленный термин *«терминал связи»* не имеет ничего общего с чем-либо специализированным, скорее всего, наоборот. Придумывая все новые и новые пути общения, мы получаем доступ ко множеству различных устройств, которые обеспечат нам возможность связи. Рассмотрим следующие сценарии:

- Если я использую персональный цифровой секретарь для соединения с голосовой почтой и получения голосовых сообщений (преобразованных в текст), становится ли он телефоном?
- Если я подключаю видеокамеру к ПК, соединяюсь с веб-сайтом компании и посылаю запрос на начало чата с сотрудником службы работы с клиентами, мой ПК стал телефоном?
- Если я используют IP-телефон на кухне для поиска рецептов в Интернете, это можно считать телефонным звонком?

Идея проста: мы, наверное, никогда не перестанем «звонить» друг другу, но всегда ли мы будем использовать «телефоны» для этого?

## Некоторые вопросы Linux

Спросите кого угодно из Free Software Foundation – и он скажет следующее: то, что мы знаем как Linux, на самом деле – GNU/Linux. Если отбросить все этимологические аргументы, в этом есть доля истины. Тогда как ядро операционной системы действительно называется Linux, подавляющее большинство утилит, установленных и регулярно используемых в системе Linux, на самом деле являются утилитами GNU. Linux, наверное, всего на 5% Linux и на 75% – GNU, а на оставшиеся 20% – пожалуй, все остальное.

Почему это важно? Гибкость Linux – одновременно и благословение, и проклятие. Благословение – потому, что с Linux можно действительно создать абсолютно индивидуальную операционную систему с нуля. Но очень немногим удалось осуществить это, поэтому в значительной степени Linux – проклятие из-за ответственности, которая ложится на наши плечи при определении, какую из утилит GNU установить и как конфигурировать систему.

Если это кажется вам неосуществимым, не пугайтесь. В следующей главе обсуждается выбор, установка и конфигурация программной среды для системы Asterisk.

## Заключение

В данной главе были рассмотрены все проблемы, которые могут повлиять на стабильность и качество установки Asterisk. Прежде чем вы совсем испугаетесь, хочется отметить, что многие люди установили Asterisk поверх рабочей станции Linux с графической оболочкой – являющейся веб-сервером, базой данных или чем-нибудь еще – вообще без всяких проблем[1]. То, сколько времени и сил придется потратить на освоение лучших практик и подсказок по проектированию, представленных в данной главе, зависит исключительно от того, насколько активно предполагается загружать сервер Asterisk и какие качество и надежность должна обеспечивать система. Если вы всего лишь экспериментируете с Asterisk, не надо волноваться слишком сильно; просто знайте, что любые возникающие проблемы, возможно, не являются недостатком системы Asterisk.

В данной главе мы попытались продемонстрировать лучшие практики, которые помогут обеспечить системе Asterisk надежную, стабильную платформу. Asterisk готова работать в намного более неблагоприятных условиях, но стабильность офисной АТС напрямую зависит от количества усилий и внимания, которые были потрачены на решение рассмотренных здесь вопросов при ее проектировании. Решение должно приниматься исходя из того, насколько ответственной будет создаваемая система Asterisk.

---

[1] Только ни в коем случае не устанавливайте среду X-Windows (которой является все, что предоставляет рабочий стол, например GNOME, KDE и т. п.). Практически гарантированно возникнут проблемы с качеством аудиосигнала, потому что Asterisk и GUI будут конкурировать в контроле над ЦП.

# 3

# Установка Asterisk

*Я жажду решать большие и выдающиеся задачи, но представлять скромные задачи так, как будто они большие и выдающиеся, является обязанностью моего шефа. Мир продвигается вперед не только мощными рывками его героев, но и скромными усилиями всех честных тружеников.*

– Хелен Келлер

В предыдущей главе мы обсудили подготовку системы к установке Asterisk. Пора браться за дело!

Asterisk можно устанавливать на многих дистрибутивах Linux[1] и различных архитектурах ПК, но в данной книге было решено сосредоточиться на одном продукте, чтобы избежать путаницы и неясностей. Мы сделали рекомендации максимально универсальными, но все равно можно заметить тяготение к структуре папок и системе утилит CentOS. CentOS (вероятно, самый популярный дистрибутив, используемый с Asterisk) был выбран потому, что его набор команд, структура папок и пр. хорошо знакомы большему числу читателей (мы обнаружили, что многие администраторы Linux знают CentOS, даже если предпочитают другой дистрибутив). Это не означает, что CentOS является единственным или даже лучшим выбором. В рассылках часто задают вопрос: «Какой дистрибутив Linux лучше всего использовать с Asterisk?» Все многообразие ответов обычно сводится к следующему: «Тот, который вам больше нравится»[2].

---

[1] А также и некоторых операционных систем не-Linux, таких как Solaris, *BSD и Mac OS X. Но следует отметить, что, хотя кому-то удавалось успешно запустить Asterisk на этих альтернативных платформах, Asterisk предназначена и продолжает активно разрабатываться для Linux.

# Какие нужны пакеты

Большинство конфигураций Asterisk включают три основных пакета: основная программа Asterisk (asterisk), драйверы телефонии (zaptel) и PRI-библиотеки (libpri). Если планируется исключительно VoIP-сеть, единственным обязательным пакетом является asterisk, но мы рекомендуем устанавливать все три пакета; какие модули активировать – можно выбрать позже. Драйверы zaptel необходимы, если используется аналоговое или цифровое оборудование или если источником временных интервалов служит драйвер ztdummy (обсуждается в данной главе позже). Библиотека libpri обязательна, только если используются PRI-интерфейсы ISDN. Можно не загружать эту библиотеку в оперативную память и сохранить небольшой объем свободного места, но мы рекомендуем установить ее вместе с пакетом zaptel для полноты.

В первом издании данной книги рекомендовалось устанавливать дополнительный пакет asterisk-sounds. Это был отдельный архив, который надо было скачать, извлечь из архива и затем установить. Теперь для Asterisk версии 1.4.0 существует два набора пакетов звуковых файлов: Core Sound и Extra Sound. Поскольку Asterisk поддерживает несколько разных аудиоформатов, эти пакеты доступны в различных звуковых форматах, таких как G.729 и GSM. Основанием для существования такого разнообразия форматов является обеспечение Asterisk возможности использовать тот звуковой формат, для которого требуется меньшее количество преобразований в ЦП. Например, если имеется большое количество соединений, поступающих по каналам VoIP, которые используют GSM, выгоднее иметь звуковые файлы в формате GSM. В окне выбора компонентов сборки (обсуждается в данной главе позже) можно выбрать один или более из предлагаемых типов звуковых файлов. Рекомендуем установить по крайней мере по одному типу из каждого пакета (Core Sound и Extra Sound). Поскольку в данной книге могут упоминаться некоторые файлы Extra Sound, предполагается, что установлен хотя бы один из этих форматов.

## Необходимые пакеты Linux

Для компиляции Asterisk в системе должен иметься компилятор GCC (версия 3.x или более поздняя) и все необходимые зависимости. Также для Asterisk требуется bison, программный генератор грамматического разбора, который заменяет yacc, и ncurses для обеспечения функциональности командной строки. Криптографическая библиотека в Asterisk требует наличия OpenSSL и его пакетов для разработки.

---

[2] В этой книге будет использоваться сервер CentOS Server 4.4, который мы обычно устанавливаем только с пакетом Editors. Если вы не знаете, какой дистрибутив выбрать, CentOS – превосходный вариант. CentOS можно найти на сайте *http://www.centos.org*.

Для Zaptel необходима библиотека libnewt и ее пакеты для разработки, чтобы обеспечить компиляцию программы zttool (см. в данной главе раздел «Использование ztcfg и zttool»). Если используются PRI-интерфейсы, Zaptel также требует установки пакета libpri (опять же, даже если линии PRI не используются, мы рекомендуем установить libpri вместе с zaptel).

Если пакеты Software Development устанавливаются на CentOS, все эти инструменты будут в наличии. Если вы стремитесь к порядку и желаете установить набор программ, только минимально необходимый для компиляции Asterisk и связанных с ней пакетов, обратитесь к табл. 3.1.

В следующей таблице использование ключа `-y` для приложения yum означает ответ «да» на все вопросы и обеспечит установку приложения и всех зависимостей без вывода этих вопросов на экран. Если это нежелательно, ключ `-y` должен быть опущен.

Если требуется установить сразу все вышеупомянутые пакеты, в командной строке можно указать несколько пакетов, например:

```
# yum install -y gcc ncurses-devel libtermcap-devel [...]
```

*Таблица 3.1. Список пакетов, необходимых для компиляции libpri, zaptel и asterisk*

| Имя пакета | Команда установки | Примечание | Используется программами |
|---|---|---|---|
| *GCC 3.x* | `yum install -y gcc` | Необходим для компиляции zaptel, libpri и asterisk | libpri, zaptel, asterisk |
| *ncurses-devel* | `yum install -y ncurses-devel` | Необходим для menuselect | menuselect |
| *libtermcap-devel* | `yum install -y libtermcap-devel` | Необходим для asterisk | asterisk |
| *Kernel Development Headers* | `yum install -y kernel-devel` | Необходим для компиляции zaptel | zaptel |
| *Kernel Development Headers (SMP)* | `yum install -y kernel-smp-devel` | Необходим для компиляции zaptel | zaptel |
| *GCC C++ 3.x* | `yum install -y gcc-c++` | Необходим для asterisk | asterisk |
| *OpenSSL* (необязательный) | `yum install -y openssl-devel` | Зависимость OSP, шифрование IAX2, res_crypto (поддержка RSA-ключа) | asterisk |

| Имя пакета | Команда установки | Примечание | Используется программами |
|---|---|---|---|
| *newt-devel* (необязательный) | `yum install -y newt-devel` | Зависимость zttool | zaptel |
| *zlib-devel* (необязательный) | `yum install -y zlib-devel` | Зависимость DUNDi | asterisk |
| *unixODBC; unixODBC-devel* (необязательный) | `yum install -y unixODBC-devel` | Зависимость func_odbc, cdr_odbc, res_config_odbc, res_odbc, ODBC_STORAGE | asterisk |
| *libtool* (необязательный; рекомендуемый) | `yum install -y libtool` | Зависимость связанных с ODBC модулей | asterisk |
| *GNU make* (версия 3.80 или более поздняя)* | `yum install -y make` | Необходим для компиляции zaptel и asterisk | asterisk |

* Обычной ошибкой тех, кто впервые устанавливает какой-либо дистрибутив Linux, является использование программы GNU make версии 3.79 или еще более ранней. Следует учесть, что правильно сборка Asterisk может быть выполнена только при наличии версии GNU make не ниже 3.80.

# Получение исходного кода

Лучше всего взять исходный код для Asterisk и его пакетов прямо на веб-сайте *http://www.asterisk.org* или FTP-сервере.

## Получение исходного кода Asterisk

Проще всего получить самую последнюю выпущенную версию с помощью программы wget.

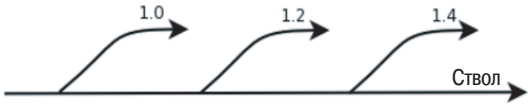
### Чем отличается код стабильной версии от кода, находящегося в процессе тестирования

Кодовая база Asterisk находится в процессе постоянного изменения. Для управления ею разработчики используют инструмент контроля версий Subversion (SVN)[1]. Subversion обеспечивает со-

[1] Subversion – превосходная система управления кодом. Ее можно найти по адресу *http://subversion.tigris.org/*. Кроме того, издательство Creative Commons выпустило не менее замечательную книгу Бена Коллинза-Сассмана (Ben Collins-Sussman) и др. «Version Control with Subversion» (O'Reilly), которая представлена по адресу *http://svnbook.red-bean.com/*.

обществу разработчиков возможность совместной работы над сложными проектами программного обеспечения.

Существует две основных области разработки Asterisk: ветвь (Branch) и ствол (Trunk). В ствол добавляются новые функции, вносятся изменения в архитектуру и всевозможные нововведения. Это та часть кодовой базы, где содержатся все новые элементы, но которая может в любой момент оказаться в нерабочем состоянии. Ее категорически нельзя использовать для производственной эксплуатации (см. рисунок).




Подобно дереву, ствол имеет ветви. Эти ветви пронумерованы соответственно основным редакциям, например 1.0, 1.2 и 1.4 (в будущем мы, скорее всего, увидим 1.6, 1.8, 1.8.2, 1.8.4. 1.8.6, 1.8.8. 1.8.8.2... м-м... и т.д...). В ветви не вносятся большие архитектурные изменения или новые функции, здесь просто исправляются дефекты и выполняются работы по обеспечению безопасности. В среде производственной эксплуатации стабильность намного важнее, чем введение новых функциональных возможностей.

Примерно каждые 14 месяцев (хотя Asterisk не следует формальному графику выпуска версий, как многие пакеты коммерческого программного обеспечения) выпускается версия Asterisk, предназначенная для использования в средах производственной эксплуатации. Начальная версия Asterisk шла под номером 1.0 и была представлена на самой первой конференции AstriCon в Атланте в сентябре 2004 года. Asterisk 1.2 была выпущена на IP4IT в ноябре 2005, а Asterisk 1.4 вышла в декабре 2006.

Обратите внимание, что извлекать из архива и компилировать исходный код Asterisk мы будем в папке /usr/src/, хотя некоторые системные администраторы, возможно, предпочитают использовать /usr/local/src. Также помните, что для записи файлов в папку /usr/src/ и установки Asterisk и сопутствующих ей пакетов, необходимо иметь права администратора.

В главе 13 можно найти информацию о том, как запустить Asterisk, не будучи администратором. Все специалисты по безопасности рекомендуют запускать свои программы-демоны под учетной записью, не дающей права администратора, на случай возможного проникновения в систему вредоносного кода. Это снизит (но, конечно же, не устранит) риск похищения пароля для пользователя с правами администратора.

Чтобы получить самую последнюю выпущенную версию исходного кода с помощью wget, в командной строке необходимо ввести следующие команды:

```
# cd /usr/src/
# wget http://downloads.digium.com/pub/asterisk/asterisk-1.4-current.tar.gz
# wget http://downloads.digium.com/pub/libpri/libpri-1.4-current.tar.gz
# wget http://downloads.digium.com/pub/zaptel/zaptel-1.4-current.tar.gz
```

Последние версии пакетов asterisk, libpri и zaptel вполне могут идти под разными номерами.

И наоборот, при разработке и тестировании, вероятно, вы захотите иметь код самой новой ветви. Чтобы извлечь этот код из SVN, выполните следующую команду:

```
# svn co http://svn.digium.com/svn/asterisk/branches/1.4 asterisk-1.4
```

Если исходный код получен в виде файлов выпущенной версии, размещенных на FTP-сервере Digium, перед компиляцией эти файлы необходимо извлечь из архива, о чем рассказывается в следующем разделе.

## Извлечение исходного кода из архива

Пакеты, загружаемые с FTP-сервера, являются архивами, в которых содержится исходный код; таким образом, перед компиляцией код требуется извлечь из архива. Если пакеты загружены в другую папку, не /usr/src/, их надо или перенести туда или указать полный путь к ним. Для извлечения исходного кода из архива мы будем использовать GNU-приложение tar. Это простой процесс, который выполняется с помощью следующих команд:

```
# cd /usr/src/
# tar zxvf zaptel-1.4-current.tar.gz
# tar zxvf libpri-1.4-current.tar.gz
# tar zxvf asterisk-1.4-current.tar.gz
```

В bash (и других командных оболочках, поддерживающих ее) можно использовать исключительно удобную функцию автозаполнения по нажатию клавиши Tab. Это позволяет вводить лишь часть имени файла, а все остальное система дополняет автоматически. Например, если введено `tar zxvf zap<tab>`, система дополнит имя файла zaptel сама. Если возможно несколько вариантов подстановки, надо нажать Tab дважды – и будет представлен список имен файлов, подходящих под введенную комбинацию символов.

Выполнение этих команд обеспечит извлечение пакетов и исходного кода из архивов в соответствующие папки. Например, файл asterisk-1.4-current.tar.gz будет извлечен в папку текущей версии Asterisk, то есть asterisk-1.4.4.

Желательно всегда сохранять исходный код последней рабочей версии пакета на случай необходимости выполнить «откат» при обнаружении очередного дефекта или странностей в поведении, которые не получается сразу разрешить.

## Окно выбора компонентов сборки

В Asterisk версии 1.4.0 и связанных с ней пакетах была реализована новая система сборки autoconf. Это немного изменило процесс сборки, но предоставило больше гибкости, позволив выбирать модули, подлежащие сборке. Преимущество состоит в том, что в сборке участвуют только необходимые модули, а не весь исходный код.

Вместе с системой сборки была введена новая система выбора на основе меню. Спасибо Расселу Брайанту (Russell Bryant). Эта новая система обеспечивает возможность более детального отбора подлежащих сборке модулей перед компиляцией ПО и избавляет пользователя от необходимости редактировать файлы Makefiles. Поэтому, чтобы не рассказывать, как использовать окно выбора компонентов сборки в каждом разделе «Компиляция...», обсудим это здесь, чтобы, увидев опцию `make menuselect`, вы знали, что должны делать с окном выбора компонентов сборки.

На рис. 3.1 представлено окно выбора компонентов сборки для программного обеспечения Asterisk. Для других пакетов оно будет прак-

```
**************************************
      Asterisk Module Selection
**************************************

         Press 'h' for help.

   ---> 1.  Applications
        2.  Call Detail Recording
        3.  Channel Drivers
        4.  Codec Translators
        5.  Format Interpreters
        6.  Dialplan Functions
        7.  PBX Modules
        8.  Resource Modules
        9.  Voicemail Build Options
        10. Compiler Flags
        11. Module Embedding
        12. Core Sound Packages
        13. Music On Hold File Packages
        14. Extras Sound Packages
```

***Рис. 3.1.*** *Пример окна выбора компонентов сборки*

тически таким же, но с меньшим количеством опций. Перемещение вверх и вниз по списку осуществляется с помощью клавиш со стрелками. Выбор опции меню выполняется по нажатию клавиши Enter или клавиши со стрелкой вправо. Для отмены выбора используется клавиша со стрелкой влево.

На рис. 3.2 показан список возможных приложений диалплана, сборка которых может быть выполнена для их использования в Asterisk. Модули, подлежащие сборке, отмечаются символами [*]. Модуль, сборка которого выполняться не будет, отмечается символами [ ]. Если перед модулем стоят символы XXX, значит, присутствуют не все зависимости пакета, которые являются обязательным условием для сборки данного модуля. На рис. 3.2 мы видим, что сборка модуля app_flash не может быть выполнена из-за отсутствия зависимости Zaptel (то есть модуль Zaptel не был собран и установлен в систему с момента последнего выполненения команды ./configure). Если требования по зависимости были удовлетворены в период после последнего выполнения команды ./configure, выполните ее снова и повторно откройте окно выбора компонентов сборки. Теперь модуль должен быть доступным для сборки.

Завершив работу с окном выбора компонентов сборки, введите символ x, чтобы сохранить изменения и закрыть окно. Ввод символа q также обеспечит выход из окна выбора компонентов сборки, но при этом изменения не будут сохранены. Если вы внесли изменения и ввели q, ваши изменения будут потеряны!

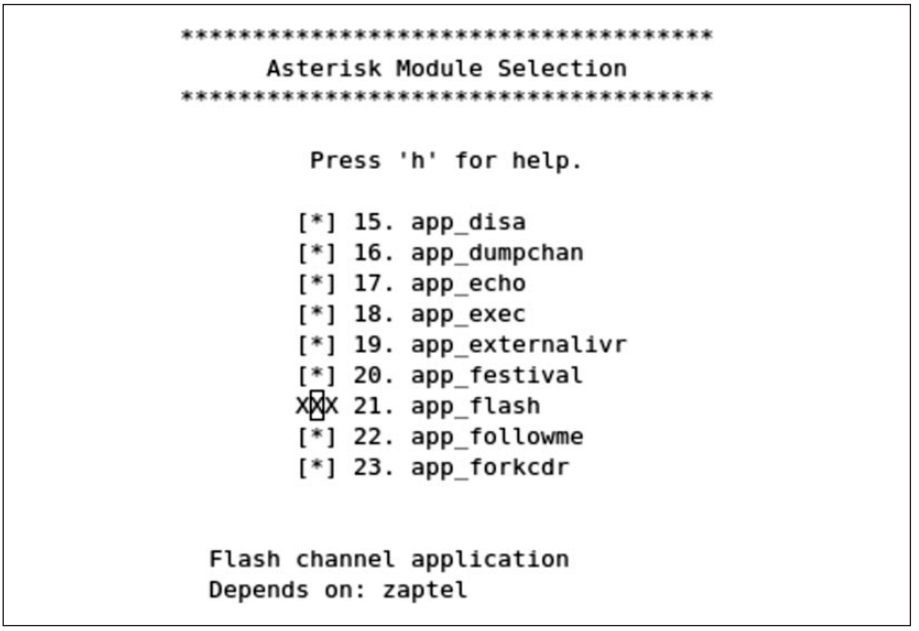



***Рис. 3.2.*** *Список модулей, подлежащих сборке*

# Компиляция Zaptel

На рис. 3.3 показаны уровни взаимодействия между Asterisk и ядром Linux с точки зрения управления аппаратными средствами. Со стороны Asterisk имеется модуль канала Zapata, chan_zap. Asterisk использует этот интерфейс для обмена информацией с ядром Linux, где загружаются драйверы устройств.

Интерфейс Zaptel – это загружаемый модуль ядра, представляющий абстрактный уровень между аппаратными драйверами и модулем Zapata в Asterisk. Именно такая концепция позволяет изменять драйверы устройств без внесения изменения в исходный код Asterisk. Драйверы устройств используются для непосредственной связи с оборудованием и для передачи информации между Zaptel и аппаратными средствами.

Хотя Asterisk может компилироваться на разных платформах, драйверы Zaptel специализированы для конкретных дистрибутивов Linux, они создаются для взаимодействия непосредственно с ядром Linux. Существует проект (*http://www.solarisvoip.com*), обеспечивающий поддержку Zaptel для Solaris. Также есть проект разработки драйверов Zapata для BSD. Его можно найти по адресу *http://www.voip-info.org/tiki-index.php?page=FreeBSD+zaptel*.

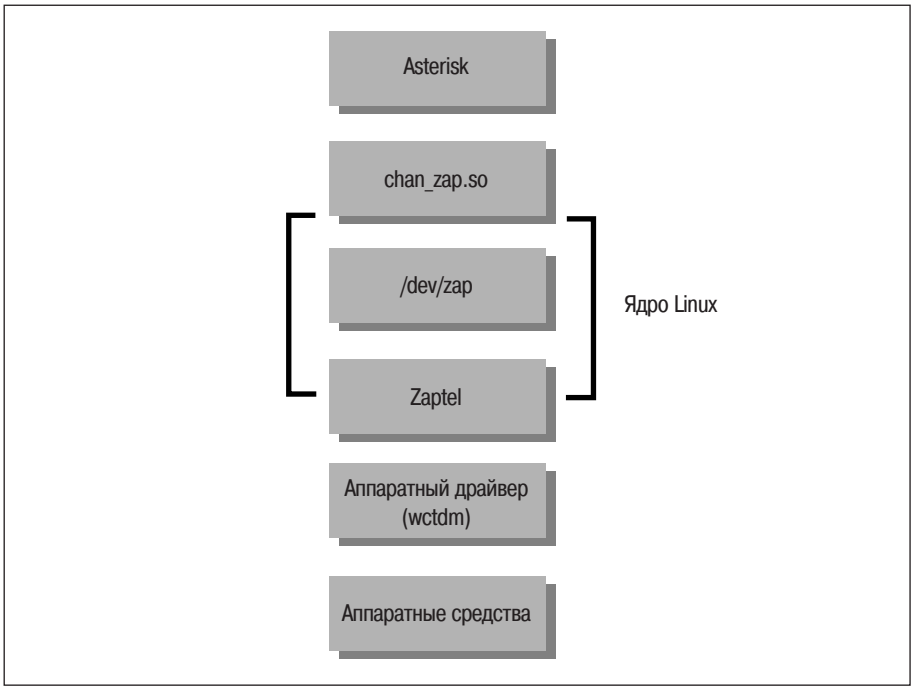



***Рис. 3.3.*** *Уровни взаимодействия устройств с Asterisk*

Начнем наше обсуждение с драйвера ztdummy, который используется в системах, нуждающихся в интерфейсе синхронизации, но не имеющих соответствующего оборудования. Затем перейдем к компиляции и установке драйверов. (Конфигурация драйверов Zaptel будет обсуждаться в следующей главе.)

Перед компиляцией драйверов Zaptel в системе, выполняющей ядро Linux 2.4, необходимо убедиться в наличии в папке /usr/src/ символической ссылки linux-2.4, указывающей на исходный код используемого ядра. Если символической ссылки нет, ее можно создать с помощью следующей команды (предполагая, что исходный код размещен в папке /usr/src/):

```
# ln -s /usr/src/'uname -r' /usr/src/linux-2.4
```

Обычно, если на компьютере выполняется один из дистрибутивов на базе ядра Linux 2.6, символическая ссылка не требуется, поскольку эти дистрибутивы осуществляют поиск папки сборки ядра автоматически. Однако, если папка сборки помещена в нестандартное место (то есть не в папку /lib/modules/<версия ядра>/build/), потребуется использовать символическую ссылку.

Хотя Asterisk и сопутствующие пакеты выполняются на ядрах Linux 2.4.x, их разработка ведется прежде всего на ядрах 2.6.x, и нет никакой гарантии, что ядра 2.4.x будут поддерживаться в будущем.

## Драйвер ztdummy

Определенные приложения и функции Asterisk для работы требуют наличия устройства синхронизации (Asterisk даже не скомпилирует их, если не найдет такое устройство). Все PCI-устройства производства Digium обеспечивают интерфейс синхронизации с частотой 1 кГц, что соответствует этому требованию. Если необходимых для обеспечения синхронизации PCI-устройств нет, в качестве устройства синхронизации может использоваться драйвер ztdummy. В дистрибутивах на базе ядра Linux 2.4 ztdummy должен использовать синхронизирующие сигналы, обеспечиваемые контроллером UHCI USB.

Многие более старые (и некоторые новые) системы используют чип контроллера OHCI USB, несовместимый с ztdummy. Однако, если используется ядро 2.6, неважно, какой чип контроллера USB применяется в системе.

Драйвер проверяет, загружен ли модуль usb-uhci и не является ли версия ядра ниже 2.4.5. Более старые версии ядра несовместимы с ztdummy.

В дистрибутиве на базе ядра 2.6 ztdummy не требует применения USB-контроллера. (Теперь ядро версии 2.6.0 генерирует синхронизирую-

щие сигналы частотой 1 кГц[1], которые драйвер может использовать для согласования; таким образом, аппаратный USB-контроллер больше не нужен.)

## Драйверы телефонии Zapata

Компиляция драйверов телефонии Zapata для использования с оборудованием Digium проста; однако из-за различия сред сборки в версиях 1.2 и 1.4 для этого применяются немного разные методы. Сначала необходимо выполнить команду `./configure`, чтобы определить, какие приложения и библиотеки установлены в системе. Это позволит гарантировать наличие всего необходимого для сборки Zaptel. Выполнение следующих команд обеспечит сборку Zaptel и его модулей:

```
# cd /usr/src/zaptel-version

# make clean
# ./configure
# make menuselect
# make
# make install
```

Команда `make clean` не всегда является обязательной, но лучше выполнять ее перед повторной компиляцией любых модулей, поскольку она удалит скомпилированные двоичные файлы из папки исходного кода. Также ее можно использовать для очистки после установки, чтобы избавиться от двоичных файлов. Обратите внимание, что выполнение данной команды обеспечит удаление двоичных файлов только из папки исходного кода, не из системы.

Кроме исполняемых файлов, `make clean` удаляет также промежуточные файлы (то есть объектные файлы) после компиляции. Они не нужны и только занимают место на жестком диске.

В системе, в которой используются папки /etc/rc.d/init.d/ или /etc/init.d/ (это такие системы, как CentOS и другие дистрибутивы на базе Red Hat), можно выполнить также команду `make config`. Это обеспечит установку сценариев запуска и конфигурирование системы. С помощью команды `chkconfig` задается автоматическая загрузка модуля zaptel при запуске:

```
# make config
```

[1] Обратите внимание, что эта характеристика ядра является настраиваемой, поэтому возможно, что некоторые дистрибутивы будут обеспечивать другую частоту, не 1000 Гц; однако в CentOS установлена именно эта частота.

Эквивалент команды `chkconfig` в Debian — `update-rc.d`.

Хотя Digium официально поддерживает только Zaptel для Linux, следует упомянуть несколько проектов по переносу Zaptel на другие платформы:

- Solaris (*http://www.solarisvoip.com*).
- BSD (*http://lists.digium.com/mailman/listinfo/asterisk-bsd*).

## Использование ztcfg и zttool

Вместе с Zaptel устанавливаются еще две программы, ztcfg и zttool. Программа ztcfg считывает данные из папки /etc/zaptel.conf для конфигурации аппаратных средств. Программа zttool поможет проверить статус установленного оборудования. Например, если используется плата T1 и между конечными точками не установлена связь, пользователь увидит красный предупреждающий сигнал. Если все сконфигурировано правильно и связь возможна, будет выведено сообщение «OK». Приложение zttool также полезно для аналоговых плат, потому что сообщает об их текущем состоянии (сконфигурирована, с подключенной линией и т. д.). Использование этих программ будет подробно рассмотрено в следующей главе.

Компиляция zttool невозможна, если не установлены библиотеки libnewt и их пакеты для разработки (newt-devel в дистрибутивах на базе Red Hat).

Приложения ztcfg и zttool и другие полезные утилиты располагаются в разделе Utilities (Утилиты) окна выбора компонентов сборки Zaptel.

# Компиляция libpri

Для библиотек libpri не используется программа autoconf для настройки среды сборки или окно выбора компонентов сборки, поскольку они не нужны; таким образом, установка упрощается. libpri применяется различными производителями аппаратных средств мультиплексирования с разделением по времени (Time Division Multiplexing, TDM), но даже если такое оборудование не установлено, эту библиотеку можно компилировать и устанавливать. libpri должна быть скомпилирована и установлена перед установкой Asterisk, поскольку она используется при его компиляции. Вот необходимые команды (вместо *версия* необходимо указать используемую версию libpri):

```
# cd /usr/src/libpri-версия

# make clean
# make
# make install
```

# Компиляция Asterisk

После компиляции и установки пакетов zaptel и libpri (если они нужны), можно переходить к установке Asterisk. В этом разделе рассматривается стандартная установка и представлены некоторые альтернативные аргументы `make`, которые могут пригодиться.

## Стандартная установка

Компиляция Asterisk выполняется с помощью компилятора gcc посредством использования GNU-программы `make`. Чтобы начать компиляцию Asterisk, просто выполните следующие команды (вместо *версия* необходимо указать используемую версию Asterisk).

```
# cd /usr/src/asterisk-версия

# make clean
# ./configure
# make menuselect
# make install
# make samples
```

Помните, что время компиляции в разных системах может быть различным. На процессорах современного поколения это не должно занять более пяти минут. На сайте AstriCon (*http://www.astricon.net*) есть сообщение об успешной компиляции Asterisk на процессоре Pentium с частотой 133 МГц, но это заняло около пяти часов. Считайте сами.

Выполнение команды `make samples` обеспечивает установку стандартных конфигурационных файлов. Ее использование (вместо конфигурации каждого файла вручную) позволит намного быстрее установить и подготовить систему Asterisk к работе. Многие значения по умолчанию нет необходимости изменять, они обеспечивают нормальную работу Asterisk. Файлы, требующие редактирования, будут рассмотрены в следующих главах.

Если в папке /etc/asterisk/ уже есть конфигурационные файлы, при выполнении команды `make samples` к имени каждого из них будет добавлено в конце расширение .old, например, файл extensions.conf будет переименован в extensions.conf.old. Однако будьте осторожны, потому что повторное выполнение команды `make samples` приведет к перезаписи исходных конфигурационных файлов!

Образцы конфигурационных файлов также можно найти в подпапке configs/ папки Asterisk sources.

Для систем, которые используют папки /etc/rc.d/init.d/ или /etc/init.d/, также желательно выполнить команду `make config`. Это обеспечит уста-

новку сценариев запуска и конфигурацию системы (с помощью команды `chkconfig`) для автоматического выполнения Asterisk при запуске:

```
# make config
```

## Альтернативные аргументы make

Существует еще несколько дополнительных аргументов `make`, которые могут передаваться во время компиляции. Некоторые из них обсуждаются здесь, но остальные используются внутри файла и на самом деле не имеют никакого значения или практической пользы для конечного пользователя. (Конечно, могут быть введены новые функции, поэтому не забывайте просматривать Makefile.)

Давайте рассмотрим некоторые полезные аргументы `make`.

### make clean

Команда `make clean` используется для удаления скомпилированных двоичных файлов из папки исходного кода. Эта команда должна выполняться перед повторной компиляцией или если требуется удалить некоторые файлы в случае недостатка места на жестком диске.

### make distclean

Команда `make distclean` используется для удаления скомпилированных двоичных файлов и для приведения папки исходного кода в начальное состояние, в каком она была после извлечения из архива.

### make update

Команда `make update` используется для замены существующего кода на обновленный код с SVN-сервера Digium. Если исходный код был загружен с FTP-сервера, будет получено уведомление об этом.

### make webvmail

Сценарий Asterisk Web Voicemail используется для предоставления графического интерфейса, предназначенного для управления учетной записью голосовой почты, что позволяет взаимодействовать с голосовой почтой удаленно с веб-броузера.

При выполнении команды `make webvmail` в папку cgi-bin/ вашего HTTP-сервера будет помещен сценарий Asterisk Web Voicemail. Если имеются специальные политики безопасности, необходимо учитывать, что эта программа использует сценарий на Perl `setuid root`. Установка будет выполнена только в CentOS или Fedora, поскольку в других дистрибутивах путь к папкам cgi-bin/ может быть иным. (Конечно, это можно изменить. На момент написания данной книги для этого необходимо было отредактировать переменную `HTTP_CFGDIR` в строке 133 Makefile.)

### make progdocs

По команде `make progdocs` с помощью программы doxygen из комментариев, внесенных в исходный код разработчиком, будет создана документация. Чтобы это было возможным, в системе должна быть установлена соответствующая программа doxygen. Заметьте, doxygen предполагает, что исходный код хорошо документирован, а это, к сожалению, не всегда так, хотя за последнюю пару лет было опубликовано очень много материалов по этому вопросу! Информация, содержащаяся в системе doxygen, будет полезна только разработчикам.

### make config

По команде `make config` в папки /etc/rc.d/init.d или /etc/init.d, если таковые будут обнаружены, устанавливаются сценарии запуска в стиле дистрибутива Red Hat. Если эти папки существуют, сценарии устанавливаются с правами доступа к файлам `755`. Если сценарий определяет, что папка /etc/rc.d/init.d/ существует, выполняется также команда `chkconfig --add asterisk`, по которой Asterisk будет добавлена в список автозагрузки. Однако в дистрибутивах, использующих только папку /etc/init.d/, этого сделано не будет. Выполнение команды `make config` никак не повлияет на уже запущенный процесс Asterisk, но запустит его, если он еще не выполняется.

В настоящее время этот сценарий полезен только в системе на базе Red Hat, хотя в папке ./contrib./init.d/ папки исходного кода Asterisk можно найти сценарии запуска и для других дистрибутивов (таких, как Gentoo, Mandrake и Slackware).

## Использование предварительно скомпилированных двоичных файлов

Задокументированный процесс установки Asterisk предполагает, что пользователь самостоятельно компилирует исходный код. Однако некоторые дистрибутивы Linux (такие, как Debian) включают предварительно скомпилированные двоичные файлы Asterisk. При наличии таких двоичных файлов пользователи могли бы устанавливать Asterisk с помощью диспетчеров пакетов, которые предоставляются такими дистрибутивами (например, apt-get для Debian и portage для Gentoo[1]). Но можно заметить, что многие из этих предварительно скомпилированных двоичных файлов довольно устаревшие и не соответствуют новейшим разработкам Asterisk.

[1] Gentoo на самом деле не использует предварительно скомпилированные двоичные файлы, а извлекает исходный код из хранилища и выполняет сборку и установку ПО с помощью собственной системы управления пакетами. Но получаемая в результате версия ПО по-прежнему зависит от того, какой производитель ее скомпоновал. А ведь все можно сделать самостоятельно!

Наконец, действительно существуют основные предварительно скомпилированные двоичные файлы Asterisk, которые можно загрузить и установить на любом выбранном дистрибутиве Linux. Однако применение таких двоичных файлов на самом деле не сэкономит много времени. Кроме того, мы пришли к выводу, что компиляция Asterisk при каждой установке не является слишком обременительной задачей. Мы верим, что лучший способ установки Asterisk – компиляция из исходного кода, поэтому в данной книге не уделяется много внимания предварительно скомпилированным двоичным файлам. Кроме того, разве вы не хотите быть 133[1]? В следующей главе будет рассмотрено, как с нуля сконфигурировать Asterisk и несколько типов каналов.

## Установка дополнительных голосовых сообщений

Дополнительные голосовые сообщения устанавливаются с помощью приложения menuselect в папку исходного кода Asterisk. Существует три набора аудиопакетов: Core Sound (Основные звуки), Extra Sound (Дополнительные звуки) и Music On Hold File (Музыка при ожидании). В каждый набор пакетов включены аудиофайлы в разных форматах (и пакеты Core Sound доступны на разных языках). Используя приложение menuselect, можно выбирать комбинации аудиопакетов для среды. Доступны следующие форматы:

- WAV.
- µlaw.
- alaw.
- GSM.
- G.729.
- G.722 (широкополосный, 16-разрядный).

На момент написания данной книги пакеты Core Sound доступны на следующих языках:

- Английский.
- Испанский.
- Французский.

Для любых звуков, определенных в окне выбора компонентов сборки, при установке система загрузит соответствующие файлы с FTP-сервера Digium. Размер этих файлов варьируется в диапазоне от 2 до 27 Мб, об этом следует помнить при установке в автономном режиме или загрузке по медленным и дорогим линиям.

[1] l33 – это забавное название «элиты» на компьютерном сленге leetspeak (написание слов с заменой букв цифрами и символами). Есть занятная, прекрасно и серьезно написанная статья о leetspeak от Майкрософт, которую можно найти по адресу *http://www.microsoft.com/athome/security/children/leetspeak.mspx*.

### Другие полезные дополнения

Пакет *asterisk-addons* содержит код, обеспечивающий возможность хранения записей параметров вызовов (Call Detail Records, CDRs) в базе данных MySQL. Также в нем имеется код, с помощью которого Asterisk может воспроизводить звуковые файлы в формате MP3 (что мы рекомендуем делать только в очень мощной системе с небольшим количеством телефонов). Кому-то также может быть интересен интерпретатор, который позволяет загружать Perl-сценарии в память на время активности процесса Asterisk (что может быть очень полезно при поступлении большого количества AGI-обращений к интерпретатору Perl). В asterisk-addons помещаются программы, недостаточно тщательно разработанные для интеграции с Asterisk или те, вопросы лицензирования которых не позволяют реализовать их непосредственно в исходном коде Asterisk.

По адресу *http://ftp.digium.com/pub/asterisk/g729/* располагаются код и программа регистрации специализированного кодека G.729A. Если установлены аудиопакеты g729, Asterisk сможет связываться с устройствами, поддерживающими кодек G.729A, но не способна преобразовывать сигналы в другие кодеки, пока не получит лицензию на использование G.729A.

# Распространенные проблемы компиляции

Существует множество проблем компиляции, с которыми часто сталкиваются пользователи. Далее рассматриваются некоторые самые распространенные из них и способы их решения.

## Asterisk

Сначала давайте рассмотрим некоторые ошибки, которые могут возникнуть при выполнении сценария `configure`.

### configure: error: no acceptable C compiler found in $PATH

Если при попытке выполнения сценария `configure` возникает такая ошибка, требуется установить компилятор gcc и его зависимости:

```
configure: error: no acceptable C compiler found in $PATH
```

(configure: ошибка: не найден необходимый компилятор C в $PATH)

Для gcc необходимы следующие пакеты:

- gcc
- cpp
- glibc-headers

- glibc-devel
- glibc-kernheaders

Эти пакеты могут быть установлены вручную путем копирования файлов с диска используемого дистрибутива или посредством диспетчера пакетов yum с помощью команды `yum install gcc`.

### configure: error: C++ preprocessor "/lib/cpp" fails sanity check

Эта ошибка возникает, если в системе не обнаружен препроцессор C++. Требуется установить пакет gcc-c++ и его зависимости:

```
configure: error: C++ preprocessor "/lib/cpp" fails sanity check
```

(configure: ошибка: препроцессор C++ "/lib/cpp" не проходит проверку на готовность к работе)

Препроцессор gcc-c++ требует наличия следующих пакетов; устанавливаются они путем выполнения команды `yum install gcc-c++`:

- gcc-c++
- libstdc++-devel

### configure: error: *** termcap support not found

Следующая ошибка может возникнуть при запуске сценария `configure`, если не установлен пакет libtermcap-devel:

```
configure: error: *** termcap support not found
```

(configure: ошибка: поддержка *** termcap не выявлена)

Для компиляции Asterisk необходим следующий файл; его можно установить, выполнив команду `yum install libtermcap-devel`:

- libtermcap-devel

## Zaptel

Ошибки могут возникать и при компиляции Zaptel. Здесь представлены некоторые наиболее распространенные проблемы и способы их решения. Если ниже вы не находите ошибки, с которой столкнулись, посмотрите предыдущий раздел, может быть, вы найдете ее там.

### make: cc: Command not found

Следующее сообщение об ошибке будет получено при попытке сборки Zaptel без компилятора gcc:

```
make: cc: Command not found
make: *** [gendigits.o] Error 127
```

(make: cc: Команда не найдена

make: *** [gendigits.o] Ошибка 127)

Убедитесь, что gcc и его зависимости установлены. Больше информации можно найти в подразделе «configure: error: no acceptable C compiler found in $PATH» предыдущего раздела.

## FATAL: Module wctdm/fxs/fxo not found

Для плат TDM400P необходима PCI-шина версии 2.2. При попытке загрузить драйверы телефонии Zapata с более старой версией могут возникать следующие ошибки:

- При попытке загрузить драйвер wctdm может появиться такое сообщение об ошибке:

  ```
  FATAL: Module wctdm not found
  ```

  (ФАТАЛЬНАЯ ОШИБКА: Модуль wctdm не найден)

- При попытке загрузить драйвер wctdm или wcfxo может появиться такое сообщение об ошибке:

  ```
  ZT_CHANCONFIG failed on channel 1: No such device or address (6)
  FATAL: Module wctdm not found
  ```

  (Не удалось выполнить ZT_CHANCONFIG для канала 1: Такое устройство или адрес не обнаружены (6)

  ФАТАЛЬНАЯ ОШИБКА: Модуль wctdm не найден)

Единственный способ исправить эти ошибки – использовать более новую системную плату, которая поддерживает PCI версии 2.2:

Также эти ошибки могут возникнуть, если блок питания не подключен в разъем Molex на плате TDM400P.

## Неразрешимая символическая ссылка при загрузке ztdummy

Драйвер ztdummy требует наличия доступного контроллера UHCI USB в ядрах Linux 2.4 (USB-котроллер не является обязательным требованием для ядер Linux 2.6, потому что они способны генерировать опорный синхросигнал частотой 1 кГц). Существуют контроллеры вторичного типа, известные как OHCI-контроллеры[1], которые несовместимы с драйвером ztdummy. Если контроллер UHCI USB недоступен в ядрах Linux 2.4, возникнет следующая ошибка:

```
/lib/modules/2.4.22/misc/ztdummy.o: /lib/modules/2.4.22/misc/ztdummy.o:
unresolved symbol unlink_td
/lib/modules/2.4.22/misc/ztdummy.o: /lib/modules/2.4.22/misc/ztdummy.o:
unresolved symbol alloc_td
```

[1] Open Host Controller Interface – открытый интерфейс хост-контроллера. – *Примеч. науч. ред.*

```
/lib/modules/2.4.22/misc/ztdummy.o: /lib/modules/2.4.22/misc/ztdummy.o:
unresolved symbol delete_desc
/lib/modules/2.4.22/misc/ztdummy.o: /lib/modules/2.4.22/misc/ztdummy.o:
unresolved symbol uhci_devices
/lib/modules/2.4.22/misc/ztdummy.o: /lib/modules/2.4.22/misc/ztdummy.o:
unresolved symbol uhci_interrupt
/lib/modules/2.4.22/misc/ztdummy.o: /lib/modules/2.4.22/misc/ztdummy.o:
unresolved symbol fill_td
/lib/modules/2.4.22/misc/ztdummy.o: /lib/modules/2.4.22/misc/ztdummy.o:
unresolved symbol insert_td_horizontal
/lib/modules/2.4.22/misc/ztdummy.o: insmod /lib/modules/2.4.22/misc/
ztdummy.o failed
/lib/modules/2.4.22/misc/ztdummy.o: insmod ztdummy failed
```

Убедиться в наличии соответствующего USB-контроллера и связанных с ним драйверов можно с помощью команды `lsmod`:

```
# lsmod
Module          Size  Used by
usb_uhci        26412 0
usbcore         79040 1 [hid usb-uhci]
```

Как можно увидеть в приведенном выше примере, мы проверяем, загружены ли модули usbcore и usb_uhci. Если эти модули не загружены, необходимо убедиться, что USB в BIOS активирован и что эти модули есть в наличии.

Если драйверы USB не загружены, все равно с помощью команды `dmesg` есть возможность проверить тип имеющегося USB-контроллера:

```
# dmesg | grep -i usb
```

Наличие контроллера UHCI USB подтвердят следующие строки:

```
uhci_hcd 0000:00:04.2: new USB bus registered, assigned bus number 1
hub 1-0:1.0: USB hub found
uhci_hcd 0000:00:04.3: new USB bus registered, assigned bus number 2
hub 2-0:1.0: USB hub found
```

(uhci hcd 0000:00:04.2: зарегистрирована новая USB-шина, присвоенный номер шины – 1

hub 1-0: обнаружен концентратор USB

uhci hcd 0000:00:04.3: зарегистрирована новая USB-шина, присвоенный номер шины – 2

hub 2-0:1.0: обнаружен концентратор USB)

## Ошибки depmod во время компиляции

Если во время компиляции возникают ошибки `depmod`, вероятнее всего, отсутствует символическая ссылка на исходные файлы ядра Linux. Если исходные файлы используемого ядра Linux не установлены, необходимо скачать их в Интернете с сайта *http://kernel.org*, установить и создать символическую ссылку на /usr/src/linux-2.4. Ниже представлен пример ошибки `depmod`:

```
depmod: *** Unresolved symbols in /lib/modules/2.4.22/kernel/drivers/block/
loop.o
```

(depmod: *** Неразрешимые символы в /lib/modules/2.4.22/kernel/drivers/block/loop.o)

# Быстрая загрузка Asterisk и Zaptel

Если команда `make config` выполняется в папках исходного кода Asterisk или Zaptel, сценарии запуска, используемые для управления Asterisk или Zaptel, будут скопированы в папку /etc/rc.d/init.d/. Используя сценарии, можно упростить загрузку Asterisk и Zaptel. Эти сценарии также выполнят команду `chkconfig`, что обеспечит автоматический запуск Asterisk и Zaptel при загрузке системы. Ниже представлен пример их использования:

```
# service zaptel start
# service asterisk start
```

Каждый сценарий запуска имеет несколько опций, с помощью которых можно управлять офисной АТС или драйверами. В табл. 3.2 и 3.3 приведены команды, выполняемые сценарием (но их можно было бы вводить самостоятельно в интерфейсе командной строки (Command-Line Interface, CLI)).

*Таблица 3.2. Опции сценария запуска Asterisk*

| service asterisk <опция> | Эквивалент для ввода вручную |
|---|---|
| start | asterisk |
| stop | killproc asterisk |
| restart | stop; start |
| reload | asterisk -rx “reload” |
| status | ps aux \| grep [a]sterisk |

*Таблица 3.3. Опции сценария запуска Zaptel*

| service zaptel <опция> | Эквивалент для ввода вручную |
|---|---|
| start | modprobe zaptel; modprobe <*модуль*>; /sbin/ztcfg |
| stop | rmmod ztdummy; rmmod zaptel |
| restart | stop; start |
| reload | /sbin/ztcfg |

# Загрузка модулей Zaptel без использования сценариев

В данном разделе будет кратко рассмотрена загрузка модулей zaptel и ztdummy без сценария запуска CentOS. Если модуль zaptel будет использоваться только для модуля ztdummy, он не требует никакой кон-

фигурации. Если планируется загружать модуль ztdummy в качестве источника временных интервалов (и таким образом, в системе не будут использоваться никакие PCI-устройства), самое время загрузить оба драйвера.

## Системы, выполняющие udevd

В эпоху юности Linux системная папка /dev/ заполнялась списком устройств, с которыми система потенциально могла взаимодействовать. В то время в этом списке было примерно 18 000 устройств. Все изменилось с появлением devfs, обеспечившего динамическое создание файлов устройств, которые активно используются системой. Некоторые последние выпущенные дистрибутивы включают программу-демон udev для динамического заполнения папки /dev/ списками устройств.

Чтобы Zaptel и другие драйверы устройств могли работать с PCI-устройствами, установленными в системе, необходимо определить несколько правил. В своем любимом текстовом редакторе откройте файл правил udevd. В CentOS, например, этот файл находится по адресу /etc/udev/rules.d/50-udev.rules. В конце файла правил необходимо добавить следующие строки:

```
# Раздел для zaptel

device
   KERNEL="zapctl",      NAME="zap/ctl"
   KERNEL="zaptimer",    NAME="zap/timer"
   KERNEL="zapchannel",  NAME="zap/channel"
   KERNEL="zappseudo",   NAME="zap/pseudo"
   KERNEL="zap[0-9]*",   NAME="zap/%n"
```

Сохраните файл и перезагрузите систему, чтобы настройки вступили в действие.

Возможно, вам не придется ничего редактировать в своей системе, потому что сценарий установки Zaptel попытается определить все правила сам; однако мы приводим эти рекомендации здесь для тех систем, которые конфигурируются не автоматически.

## Загрузка Zaptel

Модуль zaptel должен быть загружен до того, как будут загружены и использованы все остальные модули. Следует отметить, что, если модуль zaptel будет использоваться с PCI-устройствами, перед его загрузкой необходимо выполнить конфигурирование в файле /etc/zaptel.conf. (Конфигурация zaptel.conf для использования с аппаратными средствами обсуждается в главе 4.) Если zaptel предназначен только для доступа к ztdummy, его можно загрузить с помощью команды `modprobe` следующим образом:

```
# modprobe zaptel
```

Если все проходит нормально, на экран ничего не выводится. Проверить успешность загрузки модуля zaptel можно с помощью команды `lsmod`. В результате ее выполнения должна быть возвращена строка с именем модуля и указанием занимаемого им объема оперативной памяти, например:

```
# lsmod | grep zaptel
zaptel               201988 0
```

## Загрузка ztdummy

Модуль ztdummy – это интерфейс устройства, обеспечивающий генерирование временных интервалов. Благодаря ему Asterisk, в свою очередь, может предоставлять временные интервалы различным приложениям и функциям, которым это необходимо. Модуль ztdummy загружается после загрузки zaptel с помощью команды `modprobe`:

```
# modprobe ztdummy
```

Если загрузка ztdummy проходит успешно, на экран ничего не выводится. Проверить, что ztdummy загружен и используется модулем zaptel, можно с помощью команды `lsmod`. Следующий вывод получен на компьютере, выполняющем ядро 2.6:

```
# lsmod | grep ztdummy
    Module                  Size    Used by
    ztdummy                 3796    0
    zaptel                201988    1 ztdummy
```

На компьютере, на котором выполняется ядро 2.4, в выводе, полученном в результате выполнения `lsmod`, будет показано, что ztdummy использует модуль usb-uhci:

```
# lsmod | grep ztdummy
    Module                  Size    Used by
    ztdummy                 3796    0
    zaptel                201988    0 ztdummy
    usb-uhci               24524    0 ztdummy
```

# Загрузка libpri без использования сценария

Библиотеки libpri не требуется загружать, как модули. Asterisk ищет libpri во время компиляции и, если находит их, конфигурируется на использование библиотек самостоятельно.

# Запуск Asterisk без использования сценариев

Asterisk может быть загружена по-разному. Самый простой способ – выполнение двоичного файла прямо из интерфейса командной строки

Linux. Точно так же Asterisk можно запускать и перезапускать и в системе, использующей сценарии `init.d`. Однако предпочтительнее запускать Asterisk с помощью сценария `safe_asterisk`.

## Команды консоли

Двоичный файл Asterisk по умолчанию располагается по адресу /usr/sbin/asterisk. Если запустить файл /usr/sbin/asterisk, Asterisk загрузится как программа-демон. Существует также несколько ключей, которые позволяют (повторно) запускать CLI Asterisk, задавать степень детальности вывода CLI и создавать дампы ядра в случае сбоя Asterisk (для отладки с помощью gdb). Чтобы увидеть все возможные опции, запустите Asterisk с ключом `-h`:

```
# /usr/sbin/asterisk -h
```

Вот список наиболее часто используемых опций:

`-c`

Консоль. Эта опция обеспечит запуск Asterisk как пользовательского процесса (не сервера) и предоставит окно командной строки Asterisk. Она пригодится при настройке параметров запуска, но не должна использоваться при нормальной работе системы (если Asterisk уже выполняется, эта опция не сработает и будет выдано сообщение об ошибке).

`-v`

Детальность сообщений. Используется для определения степени детальности сообщений, выводимых при отладке с помощью CLI. Чем выше значение `v`, тем более детальными будут сообщения.

`-g`

Дамп ядра. При неожиданном сбое Asterisk этот ключ обусловил бы создание файла ядра для последующей его трассировки с помощью gdb. Обычно этот ключ не используется в производственной эксплуатации, а применяется только в том случае, если пишется код для Asterisk и требуется отладить все возникающие сбои.

`-r`

Удаленный. Используется для удаленного повторного подключения к уже выполняющемуся процессу Asterisk. (Процесс является удаленным с позиции консоли, подключающейся к нему, но фактически процесс выполняется локально на том же компьютере. Это не имеет ничего общего с подключением к удаленному процессу по сети с использованием, например, протокола IP, поскольку такое взаимодействие не поддерживается.) Это самая распространенная опция. Именно она использовалась бы для соединения с Asterisk в системе, в которой она выполняется как программа-демон/сервис, запущенная в момент запуска системы.

`-x "<команда CLI>"`

Выполнить. Использование этой команды в сочетании с опцией `-r` позволяет выполнять команду CLI без необходимости подключения к CLI и ввода команды вручную. В качестве примера можно привести команду на повторный запуск, для инициации которого пришлось бы ввести в командной строке `asterisk -rx "reload"`.

Рассмотрим некоторые примеры. Запустить Asterisk как пользовательскую программу (потому что выполняется настройка конфигурационного файла и придется запускать и останавливать выполнение несколько раз) и задать уровень детальности сообщений 3 можно, используя следующую команду:

```
# /usr/sbin/asterisk -cvvv
```

Если процесс Asterisk уже выполняется (например, если Asterisk сконфигурирована как часть процесса запуска системы), используется ключ восстановления соединения:

```
# /usr/sbin/asterisk -vvvr
```

Если требуется, чтобы Asterisk выводила файл ядра после сбоя, при запуске Asterisk можно использовать ключ `-g`:

```
# /usr/sbin/asterisk -g
```

Чтобы выполнить команду, не подключаясь к CLI и не вводя ее (возможно, для того, чтобы использовать ее в сценарии), можно применять ключ `-x` в сочетании с ключом `-r`:

```
# /usr/sbin/asterisk -rx "restart now"
# /usr/sbin/asterisk -rx "database show"
# /usr/sbin/asterisk -rx "sip show peers"
```

Если возникают сбои и хотелось бы записывать информацию в файл отладки, используется следующая команда:

```
# /usr/sbin/asterisk -vvvvc | tee /tmp/debug.log
```

Обратите внимание, что ключ `v` не нужен, если не требуется, чтобы система предоставляла подробную информацию о происходящем. В сильно загруженной системе этот вывод может перекрывать другую выводимую в консоли информацию.

## Папки, используемые Asterisk

Asterisk использует несколько папок в системе Linux для организации различных аспектов системы, таких как запись сообщений голосовой почты, голосовые сообщения и конфигурационные файлы. В данном разделе обсуждаются необходимые папки. Все они создаются во время установки и конфигурируются в файле asterisk.conf.

## /etc/asterisk/

В папке /etc/asterisk/ располагаются конфигурационные файлы Asterisk. Однако один файл, zaptel.conf, находится в папке /etc/. Аппаратные средства Zaptel изначально были разработаны Джимом Диксоном (Jim Dixon), сотрудником компании Zapata Telephony Group, как подходящее и доступное по цене оборудование для компьютерной телефонии. Asterisk использует это оборудование, но любое другое ПО также может воспользоваться устройствами и драйверами Zaptel. Поэтому конфигурационный файл zaptel.conf вынесен из папки /etc/asterisk/.

## /usr/lib/asterisk/modules/

В папке /usr/lib/asterisk/modules/ располагаются все загружаемые модули Asterisk. В этой папке находятся различные приложения, кодеки, форматы и каналы, используемые Asterisk. По умолчанию Asterisk загружает все эти модули при запуске системы. Любые неиспользуемые модули можно отключить в файле modules.conf, но при этом необходимо помнить, что некоторые модули необходимы Asterisk или являются зависимостями других модулей. Попытка загрузить Asterisk без этих модулей приведет к ошибке при запуске.

## /var/lib/asterisk

В папке /var/lib/asterisk/ находится файл astdb и ряд подпапок. Файл astdb содержит информацию локальной базы данных Asterisk, что несколько напоминает реестр Microsoft Windows. База данных Asterisk – простая реализация на базе версии 1 Berkeley BD. Один из исходных файлов Asterisk, db.c, информирует, что эта версия была выбрана по следующей причине: «Реализация DB3 создана по альтернативной лицензии, несовместимой с общедоступной лицензией (General Public License, GPL). Таким образом, чтобы не усложнять лицензирование Asterisk, было решено использовать версию 1, поскольку она выпущена по лицензии BSD».

Папка /var/lib/asterisk/ включает следующие подпапки:

*agi-bin/*

В папке agi-bin/ находятся специальные сценарии, которые могут взаимодействовать с Asterisk через различные встроенные приложения AGI. Подробнее об AGI рассказывается в главе 8.

*firmware/*

Папка firmware/ содержит встроенное ПО для различных совместимых с Asterisk устройств. В настоящее время в этой папке имеется только подпапка iax/, в которой находится двоичное отображение встроенного ПО для IAXy производства Digium.

*images/*

Приложения, которые соединяются с каналами, поддерживающими изображения, ищут папку images/. Большинство каналов не поддерживают передачу изображений, поэтому эта папка используется редко. Однако если появится больше устройств, поддерживающих и использующих изображения, эта папка станет более значимой.

*keys/*

Asterisk может использовать систему открытых/закрытых ключей для аутентификации равноправных участников сети, которые соединяются с модулем, используя цифровую подпись RSA. Если поместить открытый ключ такого участника сети в свою папку keys/, этот участник сети сможет быть аутентифицирован каналами, поддерживающими данный метод (такими, как каналы IAX2). Закрытый ключ никогда не предоставляется. Справедливо и обратное: вы можете предоставить свой открытый ключ равноправным участникам сети, что позволит вам проходить аутентификацию с использованием своего закрытого ключа. И открытый, и закрытый ключи – файлы с расширениями .pub и .key соответственно – хранятся в папке keys/.

*mohmp3/*

Если Asterisk сконфигурирована на воспроизведение музыки при ожидании, приложения, использующие эту функцию, ведут поиск файлов в формате MP3 в папке mohmp3/. Asterisk несколько требовательна к форматированию MP3-файлов, поэтому следует использовать кодирование с постоянной скоростью передачи данных (constant bitrate, CBR) и удалять теги ID3 из файлов.

*sounds/*

Все доступные для Asterisk голосовые сообщения находятся в папке sounds/. Основные сообщения, поставляемые с Asterisk, объединены в файл sounds.txt, размещающийся в папке исходного кода Asterisk. Дополнительные подсказки находятся в файле sounds-extra.txt, расположенном в той папке, в которую ранее в этой главе был извлечен из архива пакет asterisk-sounds.

## /var/spool/asterisk/

Папка Asterisk spool имеет несколько подпапок, включая dictate/, meetme/, monitor/, outgoing/, system/, tmp/ и voicemail/ (рис. 3.4). Asterisk отслеживает папку outgoing на наличие текстовых файлов, содержащих информацию запросов вызовов. Эти файлы позволяют производить вызов, просто перемещая правильно структурированный файл в папку outgoing/.

Файлы вызовов, помещенные в папку outgoing/, могут содержать полезную информацию, такую как Context (Контекст), Extension (Расширение) и Priority (Приоритетность), соответственно которой должен на-

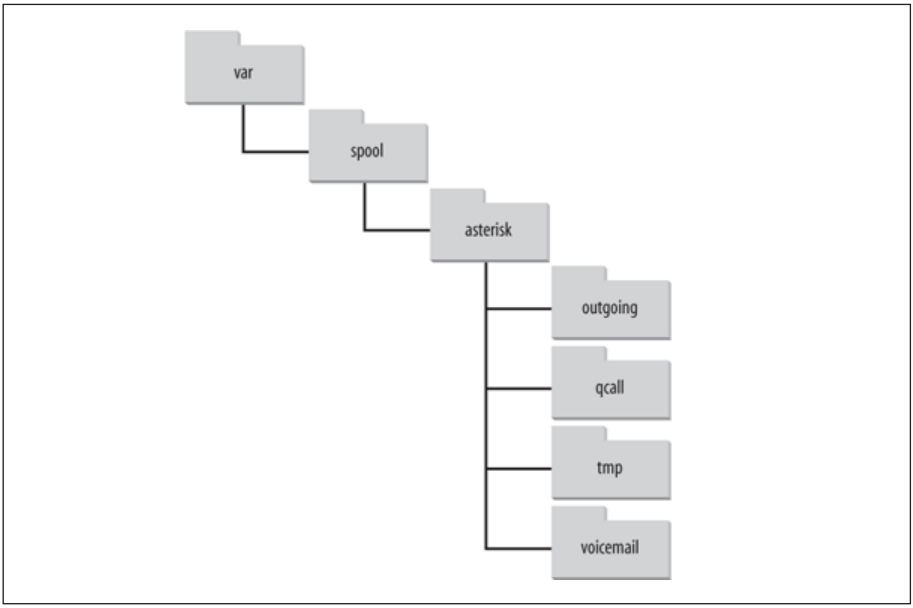



***Рис. 3.4.*** *Структура папки /var/spool/asterisk/*

чинаться ответ на вызов, или просто приложение и его аргументы. Также в них можно задать переменные и определить код учетной записи для Call Detail Records (Записи параметров вызовов). Более подробная информация об использовании файлов вызовов представлено в главе 9.

dictate/ – папка, в которой приложение `Dictate()` ищет файлы по умолчанию.

meetme/ – папка, в которой сохраняются записи конференций, организованных приложением `MeetMe()`.

Записи, полученные с помощью любого приложения для записи «в одно касание» (флаги `w` и `W` для приложения `Dial()`), `MixMonitor()` или `Monitor()`, хранятся в папке monitor/.

Папка system/ используется приложением `System()` для временного хранения данных.

Папка tmp/, как это ни смешно, используется для хранения временной информации. Некоторым приложениям может потребоваться место для записи файлов перед копированием их окончательных версий в постоянное местоположение. Это предотвращает попытки одновременной записи и чтения файла разными процессами.

Все сообщения голосовой почты и приветствия пользователя находятся в папке voicemail/. Для добавочных номеров, заданных в voicemail.conf и по крайней мере один раз зарегистрировавшихся в системе, в voicemail/ создаются подпапки.

## /var/run/

Папка /var/run/ содержит идентификаторы процессов (Process ID, PID) для всех активных процессов системы, включая Asterisk (как определено в файле asterisk.conf). Обратите внимание, что в разных ОС для этих целей могут использоваться различные папки.

## /var/log/asterisk/

Папка /var/log/asterisk/ является местом хранения журнала Asterisk. Редактируя файл logger.conf, находящийся в папке /etc/asterisk/, можно управлять типом информации, протоколируемой в различных файлах. Базовая конфигурация файла logger.conf рассматривается в приложении D.

## /var/log/asterisk/cdr-csv

Папка /var/log/asterisk/cdr-csv используется для хранения записей параметров вызовов (CDR) в формате CSV (с разделяющими запятыми). По умолчанию информация хранится в файле Master.csv, но для отдельных учетных записей с помощью опции `accountcode` можно организовать хранение их CDR в отдельных файлах (подробнее об этом рассказывается в приложении A).

# AsteriskNOW™

В следующих разделах дается последовательное введение в программный продукт AsteriskNOW, который предоставляет собой завершенную систему офисной АТС и графическое окно настройки конфигурации – все в одном!

## Что такое AsteriskNOW

AsteriskNOW – это программное устройство с открытым исходным кодом, специализированный дистрибутив Linux, включающий Asterisk, графический пользовательский интерфейс (GUI) Asterisk и все остальное программное обеспечение, необходимое системе Asterisk. С помощью GUI Asterisk можно без труда сконфигурировать систему Asterisk, даже не являясь техническим специалистом.

Примечание: полный дистрибутив программного продукта предоставляется по общедоступной лицензии (*http:// www.gnu.org/copyleft/gpl.html*) и может на законных основаниях использоваться в любых целях, в том числе и коммерческих.

## Прежде чем начать

Установка AsteriskNOW проста, потому что этот дистрибутив включает только компоненты, необходимые для выполнения, отладки и сбор-

ки Asterisk. Больше не надо думать о версиях ядра и зависимостях пакетов. AsteriskNOW – специальный дистрибутив Linux для Asterisk, созданный на базе rPath Linux.

## Что понадобится

- Система, в которой можно установить AsteriskNOW.
- Устройство для записи CD и соответствующее ПО.
- Соединение с Интернетом.
- Броузер Firefox.

В настоящее время GUI Asterisk для оптимальной производительности необходим броузер Firefox (доступен по адресу *http://www.mozilla.com/en-US/*). Более широкая поддержка броузеров будет доступна в будущих версиях.

## Установка

При подготовке и установке нового дистрибутива следует выполнять все обычные меры предосторожности. Экспресс-установка (Express Installation) удалит с жесткого диска все существующие операционные системы. Чтобы попробовать поработать с AsteriskNOW, но сохранить при этом имеющуюся систему, необходимо использовать одну из альтернативных установок (обсуждаются в разделе «Альтернативные установки»). Информация о том, где можно найти больше советов и рекомендаций по Asterisk и rPath, приведена в разделе «Дополнительные источники» в конце данной главы.

### Быстрая установка

Основная установка AsteriskNOW на самом деле довольно проста и обеспечивает возможность быстро установить и получить работоспособную систему. Эта быстрая установка может использоваться, если нет возражений против принятия настроек по умолчанию. Вся справочная информация, которая может понадобиться, представлена в окнах установки. Более детальную информацию о процедуре установки можно найти в разделе «Развернутая процедура» ниже.

1. Скачайте ISO-файл AsteriskNOW (*http://www.asterisknow.org/downloads*) и создайте из него CD-образ. Этот шаг необходим для того, чтобы можно было начать установку. Процесс создания CD-образа может быть различным в зависимости от используемого ПО для записи CD.
2. Вставьте созданный CD AsteriskNOW в устройство CD-ROM ПК.
3. Перезапустите ПК и выполните загрузку с CD. На экране появится основное меню загрузки AsteriskNOW с несколькими опциями:

- Чтобы установить или обновить систему в графическом режиме, нажмите клавишу Enter.
- Чтобы установить или обновить систему в текстовом режиме Linux, введите команду `linux text` и затем нажмите клавишу Enter.

Рекомендуется и используется по умолчанию графический режим. Если команда не была введена, установка продолжается в графическом режиме.

4. С этого момента действуйте согласно понятным без дополнительных объяснений подсказкам, появляющимся на экране, которые помогут выполнить установку.
5. По завершении установки система предложит выполнить перезагрузку. После перезагрузки на экран будет выведен URL для доступа к GUI Asterisk.
6. Теперь все готово для конфигурации и запуска AsteriskNOW.

## Развернутая процедура

1. Скачайте ISO-файл AsteriskNOW *(http://www.asterisknow.org/downloads)* и создайте из него CD-образ. Этот шаг необходим для того, чтобы можно было начать установку. Процесс создания CD-образа может быть различным в зависимости от используемого ПО для записи CD.
2. Вставьте созданный CD AsteriskNOW в устройство CD-ROM.
3. Перезапустите ПК и выполните загрузку с CD. На экране появится основное меню загрузки AsteriskNOW с несколькими опциями:
   - Чтобы установить или обновить систему в графическом режиме, нажмите клавишу Enter.
   - Чтобы установить или обновить систему в текстовом режиме Linux, введите команду `linux text` и затем нажмите Enter.

   Рекомендуется и используется по умолчанию графический режим. Если команда не была введена, установка продолжается в графическом режиме.

   После недолгой обработки на экран выводится окно установки. Первое окно выглядит следующим образом:
4. В первом окне установки можно найти сведения о версии или справочную информацию. Когда будете готовы, щелкните по кнопке Next (Далее), чтобы продолжить установку.

   В следующем окне установки производится выбор типа установки. Предлагаются два типа установки:

   Express Installation (Экспресс установка)

   Экспресс-установка обеспечит установку всего ПО, необходимого для установки Asterisk. В этом случае устанавливаются инструменты отладки и разработки.

Expert (Эксперт)

Этот тип установки следует выбирать, если вы хотите полностью контролировать все опции установки. Сюда входит выбор пакета ПО, разбиение на разделы и выбор языка.

Типом установки по умолчанию является экспресс-установка. Этот тип установки предназначен для тех, кто владеет английским языком и не желает вдаваться в тонкости процесса установки. Если вы не читаете по-английски и/или хотите контролировать детали установки, выбирайте эксперт-установку. Здесь обсуждается экспресс-установка.

5. Выберите тип установки и щелкните по кнопке Next (Далее).

На экран выводится окно Automatic Partitioning (Автоматическое разбиение на разделы). В нем предлагается несколько опций, которые необходимо определить, прежде чем ПО разобьет на разделы жесткий диск. Вы можете выбрать, какие данные (если таковые имеются) будут удалены из системы и как будет разбит на разделы жесткий диск. Предлагаются следующие варианты:

Remove All Linux Partitions (Удалить все разделы Linux)

Выбор этой опции обеспечит уничтожение всех разделов Linux, созданных при предыдущих установках Linux.

Remove All Partitions (Удалить все разделы)

Эту опцию следует выбрать, если требуется удалить все разделы системы, включая те, которые были созданы другими операционными системами (такими, как Windows).

Keep All Partitions (Сохранить все разделы)

Эту опцию следует выбрать, если требуется сохранить все текущие данные и разделы. Для реализации Asterisk потребуется достаточно пространства на жестком диске. 20 Гб – реально возможный минимум, но действительный минимальный объем зависит от требований создаваемой системы.

В большинстве случаев выбирается вариант Remove All Partitions (Удалить все разделы). Полностью отдать жесткий диск под реализацию Asterisk – лучший способ гарантировать максимальную производительность. Установите флажок Review (Просмотр) в окне Automatic Partitioning (Автоматическое разбиение на разделы), если хотите просмотреть или изменить разбиение на разделы.

6. В окне Automatic Partitioning (Автоматическое разбиение на разделы) отображается список доступных жестких дисков. Установите флажок напротив жесткого диска (или нескольких жестких дисков), который вы хотите использовать для системы. Щелкните по кнопке Next (Далее), чтобы продолжить установку.

- Если выбрана опция Remove All Partitions (Удалить все разделы) или Remove All Linux Partitions (Удалить все разделы Linux), на экране появится диалоговое окно с просьбой подтвердить желание продол-

жать установку. Щелкните по кнопке Yes (Да), чтобы продолжить, или No (Нет), чтобы изменить выбор относительно разделов.

- Если в окне Automatic Partitioning (Автоматическое разбиение на разделы) была включена опция Review (Просмотр), на экран будет выведено окно с созданными разделами. В этом окне можно изменять разделы. Чтобы продолжить, щелкните по кнопке Next (Далее).

7. На экран выводится окно Network Configuration (Конфигурация сети).
   - В окне Network Configuration (Конфигурация сети) можно сконфигурировать сетевые устройства, связанные с системой. Программа установки автоматически выявляет все сетевые устройства, подсоединенные к системе, и отображает их в списке Network Devices (Сетевые устройства). Можно принять автоматически выбранное программой установки устройство (либо несколько устройств) или редактировать выбор с помощью опции Edit (Редактировать).
   - Задайте значение Hostname (Имя хоста). Для этого выберите вариант Automatically via DHCP (Автоматически с помощью протокола DHCP) или Manually (Вручную) и введите имя хоста для своей системы. Задав имя, щелкните по кнопке Next (Далее), чтобы продолжить.
8. На экран выводится окно Time Zone Selection (Выбор часового пояса).
   - Окно Time Zone Selection (Выбор часового пояса) предлагает на выбор несколько вариантов часового пояса для вашей установки. Можно воспользоваться картой мира, на которой показаны основные города, списком районов и часовых поясов или опцией System Clock Uses UTC (Системные часы используют UTC[1]), чтобы использовать системное время. Выбрав часовой пояс, щелкните по кнопке Next (Далее).
9. На экран выводится окно Administrator Password (Пароль администратора).
   - Необходимо задать пароль для учетной записи администратора AsteriskNOW, admin. Этот пароль будет использоваться для входа в систему и GUI Asterisk. Задайте и подтвердите пароль администратора и затем щелкните по кнопке Next (Далее), чтобы продолжить.
   - На экран выводится окно About to Install (Готов начать установку), предоставляющее возможность отложить или отменить процесс установки. Если вы готовы продолжать установку, щелкните по кнопке Next (Далее).
10. На экран выводится окно Installing Packages (Устанавливаются пакеты).
    - В течение всего процесса установки AsteriskNOW на экране будет отображаться окно Installing Packages (Устанавливаются пакеты). Установка займет несколько минут.

---

[1] UTC (Universal Time Coordinated) – всемирное координированное время. – *Примеч. науч. ред.*

- По завершении установки система попросит выполнить перезагрузку. Выньте из устройства созданный вами установочный диск и щелкните по кнопке Reboot (Перезагрузка). После перезагрузки на экране появится URL для доступа к GUI Asterisk.

## Доступ к GUI

После завершения установки и перезагрузки компьютера можно переходить к GUI Asterisk. URL, используемый для доступа к GUI Asterisk, – это IP-адрес или имя хоста, которое выводится на экран после перезагрузки компьютера. Введите этот IP-адрес в адресной строке броузера. С помощью GUI Asterisk можно улучшать и настраивать AsteriskNOW.

## Альтернативные установки

AsteriskNOW можно также испытать, используя образ VMware Player (*http://www.vmware.com/download/player/*), универсальный образ гостевого домена Xen (*http:// wiki.rpath.com/wiki/Xen_Solutions_Using_rPath_Technologies*) или LiveCD (только что записанного и запущенного). Все альтернативные установки можно найти на странице для скачивания AsteriskNOW (*http://www.asterisknow.org/downloads*).

Примечание: при использовании LiveCD имя пользователя и пароль по умолчанию – admin и password соответственно.

## Дополнительные источники

В настоящее время сообществом разработчиков Asterisk на форумах Asterisk создается «Руководство пользователя AsteriskNOW». Дополнительную информацию по AsteriskNOW, включая скриншоты процесса установки по шагам и конфигурирования с помощью Мастера настройки, можно найти по адресу *http://www.asterisknow.org*. Посетите также форумы Asterisk (*http://forums.digium.com*). Больше информации и справочных данных по rPath Linux представлено в разделе Википедии rPath, *http://wiki.rpath.com*.

# Заключение

В данной главе рассмотрены процедуры получения, компиляции и установки Asterisk и связанных с ней пакетов. В следующей главе мы коснемся вопроса исходной конфигурации системы с точки зрения различных каналов связи, таких как аналоговые устройства, подключенные к портам FXS и FXO, SIP-каналы и конечные точки IAX2.

# 4

# Исходная конфигурация Asterisk

*Я не всегда понимаю, о чем говорю, но всегда уверен, что я прав.*
– Мухаммед Али

После выполнения всех этапов, описанных в главе 3, должна быть получена рабочая система Asterisk. Если что-то не так, не пожалейте времени, вернитесь назад и еще раз просмотрите все шаги, проконсультируйтесь в Википедии, привлеките сообщество и заставьте свою систему работать.

К сожалению, пока мы не можем никуда позвонить, потому что еще не создан ни один канал. Чтобы оторвать этот самолет от земли, необходима взлетная полоса. Существуют десятки разных типов каналов и различных способов сконфигурировать каждый из них, но мы просто хотим получить возможность позвонить, поэтому давайте сделаем это и не будем пока вдаваться в тонкости. Мы решили привести здесь рекомендации по конфигурации четырех каналов: Foreign eXchange Office (FXO), Foreign eXchange Station (FXS), Session Initiation Protocol (SIP) и Inter-Asterisk eXchange (IAX)[1]. Были выбраны именно эти типы, потому что, бесспорно, они самые популярные из каналов, используемых в малых системах Asterisk, а одной из принципов данной книги является разумная простота. Рассматривая эти каналы, мы не стремимся к тому, чтобы дать исчерпывающий и всесторонний обзор всех типов

[1] Официально текущей версией является IAX2, но, поскольку от поддержки IAX1 отказались много лет назад, под IAX и IAX2 подразумевается одна и та же версия.

линий или топологий, а просто представляем основную информацию, необходимую для разработки телекоммуникационной системы. Подробно конфигурация сценариев и каналов рассматривается в приложении D.

Начнем с изучения базовой конфигурации аналоговых интерфейсов, таких как порты FXS и FXO, на примере платы Digium TDM11B (которая является аналоговой платой с одним портом FXS и одним портом FXO)[1].

Далее мы коснемся нескольких интерфейсов для передачи голоса по IP-протоколу (Voice over Internet Protocol, VoIP): рассмотрим локальный канал SIP и IAX2, подключенный к программному телефону или телефонному аппарату, а также соединение двух серверов Asterisk по этим двум популярным протоколам.

Для SIP будут рассмотрены телефонные аппараты Linksys, Polycom, Aastra, Grandstream и Cisco. Приносим извинения, если в этот список не вошла используемая вами модель телефона. Однако важно то, что при всем многообразии предлагаемых этими аппаратами настраиваемых параметров обычно, чтобы устройство заработало, необходимо задать только несколько из них. Это и будет нашей целью, потому что мы считаем, что для первого раза достаточно просто привести устройство в рабочее состояние, а не выполнять полную идеальную настройку. Здесь не будут обсуждаться все функции, которые, возможно, вам хотелось бы использовать (такие, как идентификация вызывающего абонента (Caller ID) или расширенные настройки кодеков и политики безопасности), но с вашего телефона можно будет звонить и принимать звонки, а это должно заставить вас улыбнуться – хорошее состояние духа не помешает, когда мы углубимся в дебри.

Проработав эту главу, вы получите базовую систему, состоящую из нескольких полезных интерфейсов. Они обеспечат основу, необходимую для изучения файла extensions.conf (подробно обсуждаемого в главе 5), в котором хранится диалплан (фактически этот файл содержит инструкции, необходимые Asterisk для построения диалплана). Не имея аналогового оборудования, некоторые примеры будет невозможно реализовать, но все равно у вас будет сконфигурированная система, подходящая для среды, поддерживающей исключительно VoIP.

## Что мне на самом деле нужно

Символ звездочки (*) используется как символ подстановки во многих приложениях. Это хорошее имя для данной офисной АТС по многим

[1] Эта конфигурация была известна как комплект Digium Dev-lite. Далее будет подробнее рассмотрено, чем различаются FXS и FXO. Попросту говоря, эта плата предоставит один порт для подключения к традиционной аналоговой линии телефонной компании (FXO) и один порт для подключения к аналоговому телефону (FXS). Это может быть телефон любого типа, который будет работать с традиционной домашней телефонной сетью.

причинам, одна из которых – огромное число типов интерфейсов, с которыми может соединяться Asterisk. К ним относятся:

- Аналоговые интерфейсы, такие как телефонная линия и аналоговые телефоны.
- Цифровые линии, такие как линии T1 и E1.
- Протоколы VoIP, такие как SIP и IAX[1].

Asterisk не требует никакого специализированного оборудования – даже звуковой карты, – хотя естественно ожидать, что система телефонной связи должна физически соединяться с телефонной сетью. Существует множество типов канальных плат, которые обеспечивают возможность соединения Asterisk с такими устройствами, как аналоговые телефоны или PSTN-сети, но они не являются обязательным условием функционирования Asterisk. На стороне пользователя (или станции) системы может использоваться любой программный телефон Windows, Linux и других операционных систем или практически любой физический IP-телефон. То есть вопрос с телефонами системы решен. Что касается линии связи, если нет прямого подключения к центральной АТС, можно передавать звонки по Интернету с помощью провайдера сервиса VoIP.

## Работа с конфигурационными файлами интерфейсов

В данной главе будет построена конфигурация Asterisk на платформе, установленной в предыдущей главе. Для первых нескольких разделов, в которых рассматриваются каналы FXO и FXS, предполагается использование комплекта TDM11B производства Digium (который поставляется с одним FXO- и одним FXS-интерфейсом). Он обеспечит возможность подключения к аналоговой сети (FXO) и аналоговому телефону (FXS). Обратите внимание, что этот аппаратный интерфейс не является обязательным; если вы хотите настроить конфигурацию для общения только по IP-протоколу, можете пропустить раздел, посвященный конфигурации SIP.

Сама по себе выполняемая здесь конфигурация не будет иметь особенного практического значения, но она станет основой для дальнейшей работы. Будут рассмотрены следующие файлы:

*zaptel.conf*

Здесь будет выполнена низкоуровневая конфигурация аппаратного интерфейса. Будут настроены один канал FXO и один канал FXS. Это сконфигурирует драйвер для ядра Linux.

[1] ...и H.323, и SCCP, и MGCP, и UNISTIM.

*zapata.conf*

В данном файле будет выполнена настройка взаимодействия Asterisk с оборудованием. Этот файл обеспечивает конфигурацию оборудования, используемого в пользовательском процессе Asterisk, на более высоком уровне.

*extensions.conf*

Будут созданы достаточно примитивные диалпланы, но они подтвердят работоспособность системы.

*sip.conf*

Здесь будет сконфигурирован SIP-протокол.

*iax.conf*

Здесь будет выполнена конфигурация входных и выходных IAX-каналов.

Следующие разделы посвящены редактированию нескольких конфигурационных файлов. Чтобы внесенные изменения возымели действие, придется перезагрузить эти файлы. После редактирования файла zaptel.conf понадобится выполнить перезагрузку конфигурации оборудования, используя строку `/sbin/ztcfg -vv` (`-vv` можно опустить, если не нужен развернутый вывод). Изменение файла zapata.conf потребует выполнения команды `module reload` из консоли Asterisk; однако изменение методов обмена сигналами требует перезагрузки системы (команда `restart`). После редактирования файлов iax.conf и sip.conf необходимо выполнить команды `iax2 reload` и `sip reload` соответственно.

Для тестирования вновь определенных устройств необходим диалплан, посредством которого могут устанавливаться соединения. Хотя диалплан Asterisk еще не рассматривался (этим мы займемся в следующей главе), для тестирования работы, выполняемой в данной главе, необходимо создать базовый файл extensions.conf.

Создайте резервную копию файла-шаблона extensions.conf (попробуйте команду bash `mv extensions.conf extensions.conf.sample`), затем создайте пустой файл extensions.conf (используя команду bash `touch extensions.conf`) и вставьте в него следующие строки:

```
[globals]

[general]
autofallthrough=yes

[default]

[incoming_calls]

[internal]

[phones]
include => internal
```

В разделе `[general]` задано `autofallthrough=yes`, что указывает Asterisk продолжать выполнение, если все действия для добавочного номера исчерпаны. Если задать этому параметру значение `no`, Asterisk после выполнения всех предусмотренных приоритетов будет ожидать ввода. Это значение является предпочтительным, если последним приложением, выполняемым для добавочного номера, является приложение `Background()`. Если задано значение `yes` (которое стало значением по умолчанию в версии 1.4), Asterisk прервет вызов после завершения выполнения `Background()` (после воспроизведения имеющихся звуковых файлов). Чтобы заставить Asterisk ожидать ввода номера после того, как приложение завершит воспроизведение предоставляемых ему голосовых сообщений, используется приложение `WaitExten()`.

Не пугайтесь, если вышесказанное не имеет сейчас для вас особого смысла. Просто мы еще не рассматривали диалплан, приложения, приоритеты и добавочные номера (все это ожидает нас в следующей главе). Поэтому пока что просто задайте `autofallthrough=yes`. Безопаснее использовать команду `autofallthrough=yes`, поскольку мы не хотим, чтобы Asterisk простаивала без дела в ожидании ввода номера, если это не указано ей явно.

Пока что это все, но данный файл будет использоваться по ходу этой главы для построения тестового диалплана, который поможет убедиться в работоспособности всех наших устройств. Также не забудьте выполнить команду `dialplan reload` из командной строки Asterisk, чтобы внесенные изменения вступили в силу. Проверьте свои изменения, введя в командной строке команду `dialplan show`:

```
*CLI> dialplan show
[ Context 'phones', created by 'pbx_config' ]
  Include =>     'internal'                              [pbx_config]

[ Context 'internal', created by 'pbx_config' ]

[ Context 'incoming_calls', created by 'pbx_config' ]

[ Context 'default', created by 'pbx_config' ]

[ Context 'parkedcalls', created by 'res_features' ]
  '700' => 1.    Park((null))                            [res_features]

-= 1 extension (1 priority) in 5 contexts. =-
```

Контекст `parkedcalls` – это внутренний контекст Asterisk, заданный в файле features.conf.

## Настройка диалплана для выполнения тестовых вызовов

Теперь давайте подробнее остановимся на тестовом диалплане, о котором мы начали разговор в предыдущем разделе, – он позволит выполнять обратный вызов программного телефона, после того как тот будет настроен, и использовать приложение диалплана `Echo()` для тестирования двусторонней аудиосвязи. Более подробно диалпланы рассматриваются в главе 5, а пока что просто добавим выделенные курсивом строки в существующий файл extensions.conf. Мы будем использовать этот диалплан по ходу данной главы и дополнять его в определенных разделах. После ввода текста в extensions.conf диалплан необходимо перезагрузить, выполнив команду `dialplan reload` из консоли Asterisk:

```
[globals]

[general]

[default]
exten => s,1,Verbose(1|Unrouted call handler)
exten => s,n,Answer()
exten => s,n,Wait(1)
exten => s,n,Playback(tt-weasels)
exten => s,n,Hangup()

[incoming_calls]

[internal]
exten => 500,1,Verbose(1|Echo test application)
exten => 500,n,Echo()
exten => 500,n,Hangup()

[phones]
include => internal
```

## Каналы FXO и FXS

Каналы FXO и FXS отличаются друг от друга лишь тем, что один из них обеспечивает тональный сигнал готовности линии. FXO-порт не генерирует тонального сигнала, он его принимает. Самый простой пример – тональный сигнал, поставляемый телефонной компанией. FXS-порт обеспечивает и тональный сигнал, и напряжение сигнала вызова (звонка), предупреждающего пользователя о входящем вызове. Оба интерфейса обеспечивают двустороннюю связь (то есть передачу и прием в обоих направлениях одновременно)[1].

[1] Двустороннюю связь иногда называют также полнодуплексной. Полудуплексная связь – это связь, которая осуществляется одновременно только в одном направлении.

Если у вашего сервера Asterisk есть совместимый FXO-порт, в него можно подключить телефонную линию телефонной компании (или telco), что позволит Asterisk использовать эту телефонную сеть для отправки и приема телефонных звонков. Кроме того, если ваш сервер Asterisk имеет совместимый FXS-порт, в него можно подключить аналоговый телефон. Таким образом, Asterisk сможет направлять поступающие вызовы в телефон и вы будете способны использовать этот телефон для звонков куда-либо.

В конфигурации порты определяются протоколом обмена сигналами, который они используют, а не их физической сущностью. Например, физический FXO-порт будет определен в конфигурации протоколом обмена сигналами FXS, а FXS-порт – протоколом FXO. Это может сбивать с толку до тех пор, пока не станут ясны причины такого явления. Платы FX_ названы так не исходя из их физической сути, а соответственно тому, какие устройства подключаются к ним. Следовательно, плата FXS – это плата, подключаемая к станции (station). Таким образом, можно заметить, что, для того чтобы справится со своими задачами, плата FXS должна вести себя как центральная АТС и использовать протокол обмена сигналами FXO. Аналогично, плата FXO подключается к центральной АТС. Это означает, что она должна будет вести себя как станция и использовать протокол обмена сигналами FXS. Модем в компьютере – классический пример устройства FXO.

Более старая плата X100P производства компании Digium использовала набор микросхем Motorola, а плата X101P (которую Digium продавала до того, как полностью переключилась на TDM400P) базируется на наборе микросхем Ambient/Intel MD3200. Эти платы являются модемами с драйверами, которые можно использовать в качестве отдельного устройства FXO (интерфейс для телефонии не может использоваться как FXS-порт). От поддержки плат X101P отказались в пользу плат TDM-серий.

Эти платы (или их клоны) НЕЛЬЗЯ использовать в средах производственной эксплуатации. Ведь не просто так они стоят на eBay всего $10.

Платы X100P/X101P не годятся для производственной эксплуатации, потому что часто являются причиной возникновения эха и не дают возможности удаленного управления разъединением. Будьте благоразумны и не тратьте время на это оборудование. Если обратиться с вопросом об этих платах к сообществу, многие ответы будут недружелюбными. Мы вас предупредили.

## Как найти порты FXO и FXS на плате TDM400P

На рис. 4.1 представлена плата TDM400P с модулями FXS и FXO. Фото черно-белое, поэтому невозможно различить цвета, но под номером 1 –

FXS-модуль зеленого цвета, а под номером 2 – FXO-модуль, оранжево-красный. В нижнем правом углу рисунка можно увидеть разъем Molex, посредством которого плата подключается к блоку питания компьютера.

Подключение FXS-порта (зеленый модуль) к PSTN может привести к выходу из строя модуля и платы из-за подачи напряжения в систему, которая предназначена для его производства, а не потребления!

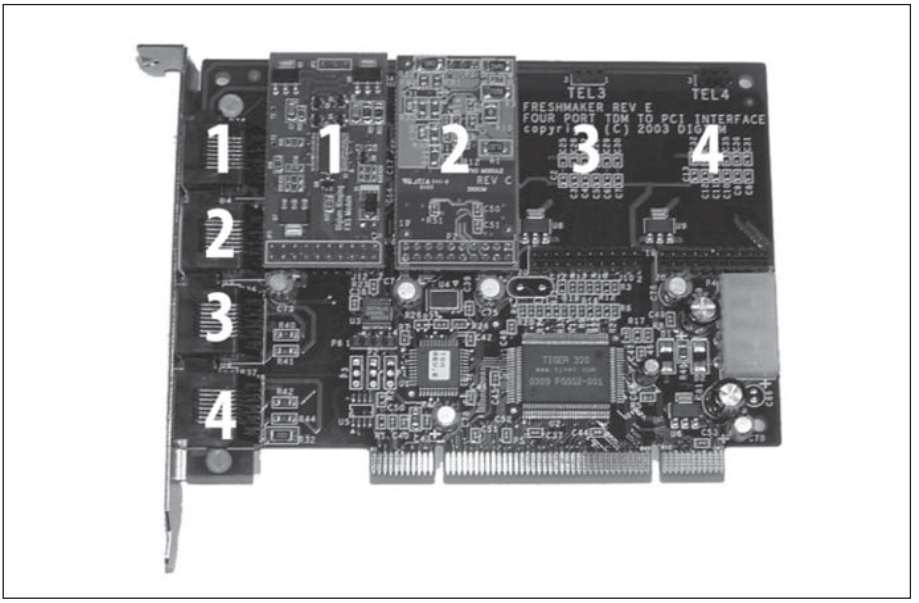



***Рис. 4.1.*** *Плата TDM400P с модулем FXS (1 по горизонтали) и модулем FXO (2 по горизонтали)*

Если на плате TDM400P есть модули FXS, убедитесь, что вы не забыли подключить питание к разъему Molex, который используется для обеспечения напряжения, необходимого для возбуждения генератора звонка в FXS-портах. Разъем Molex не нужен, если имеются только FXO-модули.

## Конфигурация канала FXO для соединения с PSTN

Начнем с конфигурации FXO-канала. Сначала сконфигурируем оборудование Zaptel, а затем – устройства Zapata. Зададим очень простой диалплан и покажем, как тестировать канал.

## Конфигурация оборудования Zaptel

Для конфигурации оборудования используется файл zaptel.conf, который находится в папке /etc/. Приведенная минимальная конфигурация определяет FXO-порт, использующий протокол обмена сигналами FXS:

```
fxsks=2
loadzone=us
defaultzone=us
```

В первой строке, кроме указания используемого протокола обмена сигналами (FXO или FXS), для канала 2 задается один из следующих протоколов:

- Loop start (`ls`)
- Ground start (`gs`)
- Kewlstart (`ks`)

Разница между loop start (кольцевая сигнализация) и ground start (сигнализация с заземлением) в том, как оборудование запрашивает тональный сигнал: линия, использующая сигнализацию с заземлением, подает сигнал дальнему концу о том, что хочет получить тональный сигнал, путем мгновенного заземления одного из проводов; линия с сигнализацией по шлейфу для запроса тонального сигнала использует замыкание. Хотя в новых системах их уже практически нет, аналоговые линии с сигнализацией с заземлением по-прежнему применяются во многих областях[1]. Сигнализация с заземлением на самом деле довольно странная штука, потому что в Asterisk нет ее аналоговой формы. Фактически передается не напряжение заземления, а сигнальный бит, предусмотренный для аналоговых сетей, присутствующих на стороне T1. Если вы чего-то здесь не понимаете, не отчаивайтесь; скорее всего, вам никогда не придется использовать сигнализацию с заземлением. Сейчас почти все линии связи (и аналоговые телефоны/модемы/факсы) используют сигнализацию по шлейфу. Kewlstart, в сущности, – это то же самое, что и кольцевая сигнализация, за исключением того что он обладает более развернутой логикой и, таким образом, может определять отсоединение удаленного конца[2]. Kewlstart является пред-

[1] Да, в многоканальных линиях T1 существует такая вещь, как сигнализация с заземлением, но она не имеет ничего общего с фактическим заземлением линии (полностью цифровой).

[2] Отсоединение удаленного конца имеет место, когда на дальнем конце линии кладут трубку. В неконтролируемой сети нет средства оповещения ближнего конца о том, что вызов завершен. Если используется телефон, это не представляет проблемы, поскольку абонент поймет, что звонок завершен, и сам повесит трубку. Однако система голосовой почты, записывающая сообщение, не имеет никакой возможности определить, что дальний конец прервал соединение, и, таким образом, будет продолжать записывать тишину, или даже тональный сигнал, или сигнал занятости линии. Kewlstart может определять такие условия и отсоединять линию.

почтительным протоколом обмена сигналами для аналоговых линий в Asterisk.

Чтобы определить другой метод обмена сигналами, не kewlstart, замените `ks` в выражении для `fxsks` на `ls` или `gs` (для кольцевой сигнализации или сигнализации с заземлением соответственно).

Параметр загрузки зоны `loadzone` задает набор сигналов (которые указаны в файле zonedata.c), используемых для канала. Файл zonedata.c содержит информацию о всевозможных звуках, воспроизводимых телефонной системой в определенной стране: тональный сигнал, сигналы дозвона, сигнал «занято» и т. д. Если применить загруженную тоновую зону к Zap-каналу, этот канал будет воспроизводить сигналы заданной страны. Для разных каналов можно задать различные наборы сигналов. Если для канала зона не задана, используется параметр зоны по умолчанию `defaultzone`.

После конфигурации zaptel.conf можно загрузить драйверы для платы. Команда `modprobe` применяется для загрузки модулей, используемых ядром Linux. Например, чтобы загрузить драйвер wctdm, нужно выполнить команду

```
# modprobe wctdm
```

Если при загрузке драйверов не выводится никаких сообщений, значит, драйверы загружены успешно[1]. Убедиться в том, что оборудование и порты были загружены и сконфигурированы правильно, можно с помощью программы ztcfg:

```
# /sbin/ztcfg -vv
```

На экран будут выведены сконфигурированные каналы и используемый метод обмена сигналами. Например, для TDM400P с одним модулем FXO будет получен такой вывод:

```
Zaptel Configuration
======================
Channel map:
Channel 02: FXS Kewlstart (Default) (Slaves: 02)
1 channels configured.
```

Если получено сообщение о следующей ошибке, для канала неправильно сконфигурирован используемый метод обмена сигналами (или отсутствует оборудование по этому адресу):

```
ZT_CHANCONFIG failed on channel 2: Invalid argument (22)
Did you forget that FXS interfaces are configured with FXO signaling and that
FXO interfaces use FXS signaling?
```

(ZT CHANCONFIG дала сбой для канала 2: Недействительный аргумент (22).

[1] Обычно в случае успешной загрузки модулей можно не просматривать отладочную информацию, но если вы хотите увидеть ее, проверьте вывод консоли (по умолчанию он находится на TTY-терминале номер 9, но это можно изменить в сценарии `safe_asterisk` – подробнее см. в предыдущей главе).

Вы забыли, что FXS-интерфейсы конфигурируются на использование протокола обмена сигналами FXO и FXO-интерфейсы используют протокол FXS?)

Чтобы выгрузить драйверы из памяти, используйте команду удаления модуля `rmmod`:

```
# rmmod wctdm
```

Программа zttool является инструментом диагностики, используемым для определения статуса оборудования. Выполнив ее, мы получаем список всех установленных аппаратных средств. В этом списке можно выбирать то или иное устройство и просматривать информацию о его текущем статусе. Статус `OK` означает, что устройство загружено успешно:

```
Alarms   Span
OK       Wildcard TDM400P REV E/F Board 1
```

## Конфигурация оборудования Zapata

Asterisk использует файл zapata.conf для определения настроек и конфигурации оборудования для телефонии, установленного в системе. Файл zapata.conf также управляет различными возможностями и функциями аппаратных каналов, такими как Caller ID, отложенный вызов, эхоподавление и многие другие.

Во время конфигурации zaptel.conf и загрузки модулей Asterisk не имеет ни малейшего представления о том, что будет конфигурироваться. Необязательно, чтобы оборудование использовалось Asterisk; его с таким же успехом могло бы применять другое ПО, взаимодействующее с модулями Zaptel. С помощью zapata.conf мы сообщаем Asterisk об аппаратных средствах и управляем соответствующими функциями:

```
[trunkgroups]
; описание всех магистральных групп

[channels]
; аппаратные каналы
; значения по умолчанию
usecallerid=yes
hidecallerid=no
callwaiting=no
threewaycalling=yes
transfer=yes
echocancel=yes
echotraining=yes

; описание каналов
context=incoming ; Используется контекст [incoming], описанный в extensions.conf
signaling=fxs_ks ; Для канала FXO используется протокол обмена сигналами FXS
channel => 2 ; PSTN подключается к порту 2
```

Раздел `[trunkgroups]` описывает соединения, в которых множество физических линий используются как одно логическое соединение с теле-

фонной сетью (они не обсуждаются в данной книге). Если вам необходим этот тип функциональности, в вашем распоряжении есть файл zapata.conf.sample и любимый поисковик для получения дополнительной информации.

Раздел `[channels]` определяет метод обмена сигналами для аппаратных каналов и их параметры. Если параметр задан, он наследуется далее по всему файлу. Для определения канала используется синтаксис `channel =>`. Каждое описание канала наследует все параметры, определенные в файле выше. Если для разных каналов требуется задать различные параметры, они должны задаваться перед описанием `channel =>`.

Строка `usecallerid=yes` служит для активации возможности Caller ID, а строка `hidecallerid=no` определяет, что идентификатор вызывающего абонента не будет скрыт для исходящих вызовов. Отложенный вызов деактивирован для FXO-линии посредством строки `callwaiting=no`. Активация вызова с подключением третьего абонента (строка `threewaycalling=yes`) позволяет переводить активный вызов в состояние ожидания путем быстрого нажатия рычажного переключателя (обсуждается в главе 7). При этом текущий вызов будет отложен. После этого можно позвонить третьей стороне и подключить ее к разговору, еще раз нажав переключатель. По умолчанию возможность вызова с подключением третьего абонента деактивирована.

Разрешение переадресации вызова с помощью рычажного переключателя настраивается строкой `transfer=yes`; для этого требуется, чтобы была активирована возможность вызова с подключением третьего абонента. Для устранения эха, которое может возникнуть в аналоговых линиях, используется эхокомпенсатор Asterisk, он активируется строкой `echocancel=yes`. Эхокомпенсатору в Asterisk требуется некоторое время на изучение эха, но этот процесс можно ускорить, включив обучение эхоподавления (`echotraining=yes`). В этом режиме Asterisk посылает тональный сигнал в линию в начале вызова, чтобы измерить эхо, и таким образом, на его изучение уходит меньше времени.

При поступлении вызова на интерфейс FXO вы захотите выполнять некоторые действия. Действия, которые будут осуществляться, задаются в блоке инструкций под названием `context`. Входящие вызовы FXO-интерфейса направляются в контекст `incoming` (входящие) посредством строки `context=incoming`. Команды, выполняемые в этом контексте, определены в файле extensions.conf.

Наконец, поскольку канал FXO использует протокол обмена сигналами FXS, это определено в строке `signaling=fxs_ks`.

## Конфигурация диалплана

Приведенный ниже простейший диалплан использует приложение `Echo()` для проверки работы двунаправленной связи в канале:

```
[incoming]
; входящие вызовы, поступающие через порт FXO,
```

```
;направляются в этот контекст из zapata.conf
exten => s,1,Answer()
exten => s,n,Echo()
```

Приложение `Echo()` будет возвращать все, что вы скажете.

## Звонок

Теперь, когда канал FXO сконфигурирован, протестируем его. Запустите приложение zttool и подключите линию PSTN к FXO-порту на плате TDM400P. Как только телефонная линия будет подключена к FXO-порту, красный световой индикатор погаснет.

Теперь наберите номер PSTN с другого внешнего телефона (например, мобильного). Asterisk ответит на вызов и выполнит приложение `Echo()`. Если вы услышали в трубке свой голос, значит, FXO-канал установлен и сконфигурирован успешно.

# Конфигурация канала FXS для аналогового телефона

Конфигурация канала FXS аналогична настройке канала FXO. Рассмотрим ее подробнее.

## Конфигурация оборудования Zaptel

Ниже представлена минимальная конфигурация для канала FXS платы TDM400P. Эта конфигурация аналогична приведенной выше конфигурации канала FXO, добавлена только строка `fxoks=1`.

Из наших предыдущих обсуждений вы должны помнить, что для каналов FXO и FXS используются противоположные типы протоколов обмена сигналами, поэтому для канала FXS мы зададим протокол FXO. В приведенном ниже примере канал 1 конфигурируется на использование протокола FXO с протоколом обмена сигналами kewlstart:

```
fxoks=1
fxsks=2
loadzone=us
defaultzone=us
```

После загрузки драйверов для своего оборудования можно проверить статус оборудования с помощью команды `/sbin/ztcfg -vv`:

```
Zaptel Configuration
======================

Channel map:

Channel 01: FXO Kewlstart (Default) (Slaves: 01)
Channel 02: FXS Kewlstart (Default) (Slaves: 02)

2 channels configured.
```

## Конфигурация оборудования Zapata

Следующая конфигурация аналогична настройке канала FXO, добавлен только раздел для FXS-порта и строка `immediate=no`. Контекстом FXS-порта является `phones`, протокол обмена сигналами – `fxoks` (kewlstart), номер канала – 1.

Каналы FXS можно сконфигурировать на выполнение одного из двух различных действий при снятии трубки телефона. Наиболее часто используемым (и обычно предпочтительным) вариантом для Asterisk является воспроизведение тонального сигнала готовности линии и ожидание ввода пользователя. Это действие конфигурируется строкой `immediate=no`. Альтернатива для Asterisk – вместо воспроизведения тонового сигнала автоматическое выполнение набора инструкций, заданных в диалплане. Такое поведение определяется строкой `immediate=yes`[1]. Инструкции, которые должны выполняться для канала, находятся в заданном для него контексте и будут соответствовать добавочному номеру `s` (обе эти темы будут обсуждаться подробнее в следующей главе).

Вот наш новый файл zapata.conf:

```
[trunkgroups]
; описание всех магистральных групп

[channels]
; аппаратные каналы
; значения по умолчанию
usecallerid=yes
hidecallerid=no
callwaiting=no
threewaycalling=yes
transfer=yes
echocancel=yes
echotraining=yes
immediate=no

; описание каналов
context=phones ; Используется контекст [internal], описанный в extensions.conf
signalling=fxo_ks ; Для канала FXS используется протокол обмена сигналами FXO


channel => 1 ; Телефон подключается к порту 1
context=incoming ; Входящие вызовы направляются в контекст [incoming],
                 ; описанный в extensions.conf
signalling=fxs_ks ; Для канала FXO используется протокол обмена сигналами FXS
channel => 2 ; PSTN подключается к порту 2
```

[1] Это также называют методом Batphone или, более формально, автоматическим прямым вызовом (Automatic Ringdown) либо автоматическим прямым вызовом частной линии (Private Line Automatic Ringdown, PLAR). Этот метод широко применяется на стойках оформления проката автомобилей и в аэропортах.

## Конфигурация диалплана

Мы воспользуемся простейшим диалпланом, сконфигурированным ранее в данной главе для тестирования FXS-порта с помощью приложения `Echo()`. Соответствующий раздел, который уже должен присутствовать в диалплане, выглядит следующим образом:

```
[internal]
exten => 500,1,Verbose(1|Echo test application)
exten => 500,n,Echo()
exten => 500,n,Hangup()

[phones]
include => internal
```

Приложение `Echo()` будет возвращать все, что вы скажете.

# Конфигурация SIP-телефонов

Протокол Session Initiation Protocol (SIP)[1], обычно применяемый в VoIP-телефонах (как аппаратных, так и программных), отвечает за установку и разъединение соединения, а также за любые изменения, происходящие во время соединения, такие как переадресации. Назначение SIP – помочь двум конечным точкам поговорить друг с другом (по возможности напрямую). Протокол SIP – это просто протокол обмена сигналами, то есть его задачей является лишь обеспечить возможность двум конечным точкам говорить друг с другом, но не работа с носителем вызова (голосом). Передача голоса осуществляется с помощью другого протокола – Real-Time Transport Protocol (транспортный протокол реального времени – RTP; RFC 3550) – для передачи медиа-данных непосредственно между двумя конечными точками.

Термин «медиа-данные» используется здесь для обозначения данных, передаваемых между конечными точками и используемых для воссоздания голоса на противоположном конце провода. Также он может использоваться в случае воспроизведения музыки или голосовых сообщений офисной АТС.

В мире SIP конечные точки называются агентами пользователя, которые могут быть двух типов: клиент и сервер. Клиент – это конечная точка, формирующая запрос, а сервер обрабатывает этот запрос и формирует ответ. Когда конечная точка желает выполнить вызов другой

[1] RFC 3261 доступен по адресу *http://www.ietf.org/rfc/rfc3261.txt*. Документ довольно объемный, но тем, кто желает стать специалистом в Asterisk, рекомендуем прочитать по крайней мере первые 100 страниц и разобраться, как устанавливать соединения, поскольку эти знания будут необходимы для работы с историей SIP (`sip debug` из консоли Asterisk) и поиска с ее помощью причины невозможности установления соединений.

конечной точки (например, наш программный телефон звонит на другой программный телефон), она формирует запрос и отправляет его на прокси-сервер SIP[1]. Прокси-сервер принимает запрос, определяет его место назначения и направляет его туда. Если два агента пользователя успешно договорились и установили вызов, переносимый сигнал передается по RTP-протоколу и пересылается непосредственно от одного агента пользователя другому. SIP-прокси не обрабатывают медиа-данные; они просто работают с SIP-пакетами.

С другой стороны, Asterisk называют Back-To-Back User Agent (B2BUA). Это означает, что Asterisk действует как агент пользователя или в роли сервера (принимающий), или в роли клиента (посылающий). Итак, когда программный телефон звонит на добавочный номер, соединение устанавливается непосредственно между программным телефоном и Asterisk. Если логика, реализованная в Asterisk, определяет, что вызов адресован другому агенту пользователя, Asterisk действует как клиент и устанавливает другое соединение (известное как канал) с другим телефоном. При этом медиа-информация передается от телефона к телефону прямо через Asterisk[2]. С точки зрения телефонов они взаимодействуют непосредственно с Asterisk.

## Базовая конфигурация SIP–телефонов в Asterisk

Конфигурация SIP-телефона для работы с Asterisk не требует много усилий и времени. Однако здесь можно легко запутаться из-за обилия опций как в Asterisk, так и в конфигурации конкретного телефонного аппарата или программного телефона. Добавьте к этому тот факт, что одни и те же вещи могут называться по-разному, – вот и прекрасный повод для того, чтобы впасть в отчаяние. Поэтому мы собираемся рассмотреть лишь самые основные вопросы. У тех, кто последует нашим советам, должно получиться заставить работать рассматриваемые здесь аппараты (а также уверенно справиться с неупомянутыми здесь телефонами). Мы не говорим, что это лучший или даже верный путь, но это самый простой путь. Намного проще взять уже работоспособную базу и настраивать ее, добиваясь необходимого решения.

Точно так же, как мы делали это для файла extensions.conf, выполните следующие команды в оболочке bash:

---

[1] OpenSER – превосходный прокси-сервер SIP с открытым исходным кодом, доступен по адресу *http://www.openser.org*.

[2] Да, существуют способы направить поток непосредственно между телефонами после того, как вызов установлен. Это можно настроить в файле sip.conf, используя или `directrtpsetup=yes` (экспериментальная опция, позволяющая перенаправлять переносимый сигнал в исходной настройке соединения), или `canreinvite=yes` (при этом переносимый сигнал проходит через Asterisk только в начале, до повторного сообщения `INVITE`, после чего сигнал может быть направлен непосредственно от телефона к телефону).

```
# mv sip.conf sip.conf.sample
# touch sip.conf
```

## Определение SIP-устройства в Asterisk

Если внести следующие строки в файл sip.conf, можно будет зарегистрировать телефон в системе.

```
[general]

[1000]
type=friend
context=phones
host=dynamic
```

Несимпатично, небезопасно, не обладает гибкостью, неполнофункционально, но это будет работать.

Даже несмотря на то что это SIP-устройство названо `1000` и, вероятно, ему будет присвоен именно этот добавочный номер, следует отметить, что имя устройства может быть произвольным. `mysipset`, `john`, `0004f201ab0c` — все эти имена действительны, широко используются и даже, возможно, больше отвечают требованиям пользователей[1]. Главное, чтобы присваиваемое имя было уникальным идентификатором устройства, который станет частью мандата при выполнении вызова по каналу SIP.

Поскольку мы хотим как направлять вызовы в программный телефон, так и обеспечить клиенту возможность размещать вызовы, параметр `type` (тип) был определен как `friend` (друг). Существует еще два параметра: `user` (пользователь) и `peer` (равноправный участник сети). С точки зрения Asterisk `user` задается для входящих вызовов, а `peer` — для исходящих звонков (через приложение `Dial()`). `friend` — это просто краткая запись, определяющая и пользователя, и равноправного участника. Если есть сомнения, задавайте тип `friend`.

Опция `host` (хост) используется для определения местонахождения клиента в сети, когда Asterisk необходимо направить ему вызов. Это значение может быть задано статически, например `host=192.168.1.100`, или, если клиент имеет динамический IP-адрес, задается `host=dynamic`. Если для опции `host` задано значение `dynamic` и клиент сконфигурирован для автоматической регистрации, Asterisk получит от конечной точки (то есть от телефонного аппарата или программного телефона) пакет `REGISTER`, из которого Asterisk сможет узнать, какой IP-адрес использует равноправный SIP-участник.

Если вы не доверяете своей сети, вероятно, следует задать пароль. Для этого в описание устройства добавляется следующая строка. Это один из тех параметров, которые не являются обязательными, но желательны:

```
secret=guessthis
```

[1] Максимальная длина имени пользователя — 255 символов.

## Конфигурация самого устройства

В меню конфигурации телефона (которые могут быть предоставлены через графический веб-интерфейс пользователя, меню самого телефона или, возможно, посредством использования конфигурационных файлов, хранящихся на сервере) уникальный идентификатор (в данном случае 1000) является составной частью мандатов, используемых для процесса аутентификации. Естественно, чтобы соединение было успешным, идентификатор в Asterisk должен совпадать с идентификатором телефонного аппарата. Забавно, что формального названия для этого идентификатора не существует. Мы решили называть его просто уникальным идентификатором.

В SIP RFC (*http://www.faqs.org/rfcs/rfc3261.html*) в разделе 19.1 этот токен пользователя назван «идентификатором конкретного ресурса хоста, к которому выполняется обращение». Эта формулировка соответствует нашему применению [1000] в качестве идентификатора аппарата в файле Asterisk sip.conf.

Вероятно, вам привычнее было бы видеть поля user name, auth name, authentication name и т. д. Здесь необходимо помнить, что на стороне Asterisk все сконфигурировано просто и правильно и поэтому можно экспериментировать с настройками телефона, пока не будет найдена работоспособная комбинация. Такой вариант намного лучше, потому что обычно новые пользователи проходят через невероятные мучения, меняя настройки и там и тут, и не могут зарегистрировать телефон.

Повторим еще раз: задайте в sip.conf максимально простую конфигурацию Asterisk и после этого не меняйте ничего. Поверьте, то, что написано здесь, будет работать. Приведите свой телефон в рабочее состояние (то есть чтобы он мог принимать и делать вызовы) и только после этого начинайте экспериментировать с разными настройками. Мы видели слишком много страданий (включая собственные) и хотим положить им конец.

### Упрощение sip.conf

Файл sip.conf (который был скопирован в папку /etc/asterisk с помощью команды make samples в предыдущей главе) содержит большое количество опций и документации, но сам файл на самом деле очень небольшой, если убрать из него все закомментированные параметры. Стандартный файл сводится всего лишь к следующим нескольким строкам, незакомментированным по умолчанию:

```
[general]
context=default    ; Контекст по умолчанию для входящих
                   ; вызовов
```

```
allowoverlap=no    ; Отключить поддержку набора номера
                   ; в режиме overlap.
                   ; (Значение по умолчанию - yes)
bindport=5060      ; Используемый UDP-порт 5060
                   ; (стандартный SIP-порт)
                   ; bindport – локальный UDP-порт,
                   ; который будет слушать Asterisk
bindaddr=0.0.0.0   ; Используемый IP-адрес 0.0.0.0
                   ; (все доступные адреса)
srvlookup=yes      ; Активировать поиск DNS SRV-записей
                   ; для исходящих вызовов
                   ; Примечание: Asterisk использует
                   ; только первый хост в SRV-записях
                   ; Деактивация поиска DNS SRV-записей
                   ; отключает возможность размещать
                   ; SIP-вызовы к другим SIP-пользователям
                   ; в Интернете на основании доменных имен
[authentication]
```

В разделе `[general]` находятся опции, которые будут применяться ко всем клиентам и каналам SIP. Некоторые настройки задаются только в разделе `[general]`, другие могут задаваться в разделе `[general]` как применяемые по умолчанию для всех условных инструкций и могут быть переопределены в другом месте. Эти опции перечислены в столбцах `[users]` и `[peers]` под заголовком `[authentication]`.

Как правило, закомментированные опции являются используемыми Asterisk по умолчанию или значение по умолчанию указано в описании опции.

Также можно проверить текущее состояние SIP-канала в Asterisk с помощью CLI-команды `sip show settings`.

Если Asterisk и программный телефон выполняются в одной системе (то есть программный телефон X-Lite и Asterisk выполняются на портативном или настольном компьютере), придется изменить SIP-порт, который слушает клиент. Его надо будет заменить с 5060 на 5061 (или любой другой свободный порт), чтобы Asterisk и программный телефон не мешали друг другу.

## Основные компоненты сервера

Перед тем как углубиться в описание следующих файлов, осталось определить еще несколько параметров на сервере. Выполнение соответствующих сервисов в собственной сети обеспечит возможность автоматической настройки аппаратов Polycom при их подключении. Работы

здесь немного, и, поверьте, результат того стоит. Потренируйтесь немного – и в будущем для любой новой системы на это будет уходить лишь несколько минут, что сохранит вам массу времени и избавит от «борьбы» с веб-интерфейсами. Когда вы возьмете новенький телефон Polycom, подключите его в свою сеть, посмотрите, как он сам себя настроит и успешно зарегистрируется на сервере Asterisk, вы испытаете наслаждение, которое доступно только истинным компьютерщикам[1].

На самом деле все не так сложно. На наш взгляд, можно запутаться только в доступных вам способах, потому что их выбор действительно огромен.

## DHCP-сервер

Обычно DHCP-сервер используется для конфигурации основных настроек протокола IP для устройства (IP-адрес, шлюз по умолчанию и DNS), но протокол DHCP (Dynamic Host Configuration Protocol – протокол динамической конфигурации хоста) на самом деле может передавать множество других параметров. В нашем случае мы хотим, чтобы он передал в телефонные аппараты некоторую информацию с указанием, откуда они могут загрузить конфигурационные файлы. Вот пример настроек DHCP-сервера Linux, которые обеспечат то, что нам требуется:

```
ddns-update-style interim;
ignore client-updates;

subnet 192.168.1.0 netmask 255.255.255.0 {
 option routers 192.168.1.1;
 option subnet-mask 255.255.255.0;
 option domain-name-servers 192.168.1.1;
 option ntp-servers pool.ntp.org;
 option time-offset -18000;

range dynamic-bootp 192.168.1.128 192.168.1.254;
default-lease-time 21600;
max-lease-time 43200;
}
```

Помните, здесь предполагается, что все устройства данной сети относятся к телефонной системе (при такой настройке IP-адрес предоставляется любому запрашивающему его устройству). Для более сложного окружения придется сконфигурировать демон DHCP для обработки различных обслуживаемых им устройств. Например, можно придумать диапазон IP-адресов, которые будут использоваться только для телефонов Polycom в локальной телефонной сети. Поскольку уникаль-

[1] Обычно это происходит в 4 часа утра после бессонных выходных перед ответственной встречей, назначенной на 8 часов утра. Red Bull, наверное, самый популярный напиток разработчиков Asterisk. Dr. Pepper – на втором месте. Кому Red Bull?

ный идентификатор организации (Organizationally Unique Identifier, OUI) всех IP-телефонов Polycom для настольных компьютеров – 00:04:f2, исходя из этого можно выбрать диапазон.

В DHCP-средах Microsoft имя tftp-сервера называют именем хоста сервера загрузки (Boot server host name). Оно определено как опция 66.

DHCP-протокол намного более гибок, чем часто кажется на первый взгляд, потому что в большинстве сред он не используется для выполнения сложных задач подготовки к работе. Приложив немного усилий и внимания, можно разработать такую среду DHCP, которая будет обслуживать и телефонные устройства, и устройства работы с данными и существенно упростит работу по администрированию при введении новых устройств.

## FTP-сервер

В настоящее время мы предпочитаем использовать для конфигурации аппаратов Polycom протокол FTP (File Transfer Protocol – протокол передачи файлов)[1]. Мы бы рекомендовали выбрать его, а не TFTP и для устройств, которые могут работать с обоими протоколами. В системе CentOS при выполнении следующей команды будет установлен VSFTPD, Very Secure FTP Daemon (очень безопасный демон FTP):

```
# yum -y install vsftpd
```

Затем для обеспечения защиты необходимо предотвратить анонимные входы в систему. Для этого вносим простое изменение в конфигурационный файл vsftpd, находящийся в папке /etc/vsftpd/vsftpd.conf:

```
# anonymous_enable=NO
```

Перезапустите сервер с помощью команды `service vsftpd restart`. Чтобы гарантировать запуск демона после каждой перезагрузки, выполняем команду `chkconfig vsftpd on`.

Теперь необходимо создать пользовательскую учетную запись и группу, которые будут использоваться для телефонных аппаратов. В данном случае мы создадим учетную запись для аппаратов Polycom.

```
# groupadd PlcmSpIp
# useradd PlcmSpIp -g PlcmSpIp -p PlcmSpIp
# passwd PlcmSpIp
```

Задаем пароль PlcmSpIp (стандартный пароль FTP для аппаратов Polycom). Его можно изменить, но после этого придется вручную на-

[1] FTP является более предпочтительным, чем TFTP, потому что телефон Polycom способен различать временны́е метки файлов FTP. Это позволяет телефону избежать повторной загрузки конфигурационных файлов и обновлений встроенных программ, которые у него уже имеются, что сокращает время загрузки.

страивать каждый аппарат, чтобы сообщить ему его нестандартные учетные данные[1].

Для большей безопасности убедимся, что FTP-сервер хранит эту учетную запись в «темнице» `chroot`:

```
# echo PlcmSpIp >> /etc/vsftpd/vsftpd.chroot_list
```

Вот примерно то, что должно быть сделано, чтобы подготовить операционную систему к предоставлению необходимых сервисов телефонам.

В следующих нескольких разделах даются рекомендации для различных популярных SIP-телефонов. Выберите самый подходящий раздел исходя из того, какой телефон вы собираетесь использовать (неважно, будет ли это аппаратный или программный телефон). Вы заметите, что всем этим телефонам мы присвоили один и тот же уникальный идентификатор. Если планируется установка нескольких телефонов, каждому из них должно быть дано уникальное имя. И не забудьте внести описания этих устройств в файл sip.conf.

## Программный телефон X-Lite производства CounterPath

Программный телефон X-Lite компании CounterPath стал очень популярным в сообществе разработчиков Asterisk. Он прост, функционален, приятен на вид и, что самое главное, распространяется бесплатно.

В этом разделе мы займемся конфигурацией программного телефона X-Lite для подключения к Asterisk. IP-адрес этого телефона – 192.168.1.250, Asterisk располагается по адресу 192.168.1.100. Доступен X-Lite для Microsoft Windows, Mac и Linux. Копию X-Lite можно скачать по адресу *http://www.counterpath.com/index.php?menu=download*.

Теперь давайте сконфигурируем программный телефон для установления соединения с сервером Asterisk. Чтобы сконфигурировать X-Lite, нажмите кнопку Settings (Настройки), которая показана на рис. 4.2.

Выберите пункты System Settings ⇒ SIP Proxy ⇒ [Default] (Настройки системы ⇒ SIP-прокси ⇒ [По умолчанию]). При этом на экран будет выведена стандартная конфигурация программного телефона. Измените настройки, как показано на рис. 4.3.

Если система Asterisk не запущена, запустите ее сейчас (информацию по установке и запуску Asterisk можно найти в главе 3). Если Asterisk выполняется в фоновом режиме, вызвать ее CLI можно, выполнив следующую команду:

```
# asterisk -rvvv
```

[1] Можно придумывать для телефонов невероятно сложные пароли, которые невозможно угадать, но если вы не собираетесь вводить их в каждый телефон вручную, их имя пользователя и пароль на FTP-сервере придется передавать с DHCP-сервера. Любое устройство, которое может регистрироваться в сети телефонной связи, способно получать информацию с DHCP-сервера. Мы не предлагаем игнорировать безопасность, просто не думайте, что создание индивидуального пароля для каждого телефона повысит ее.


***Рис. 4.2.*** *Конфигурация X-Lite*

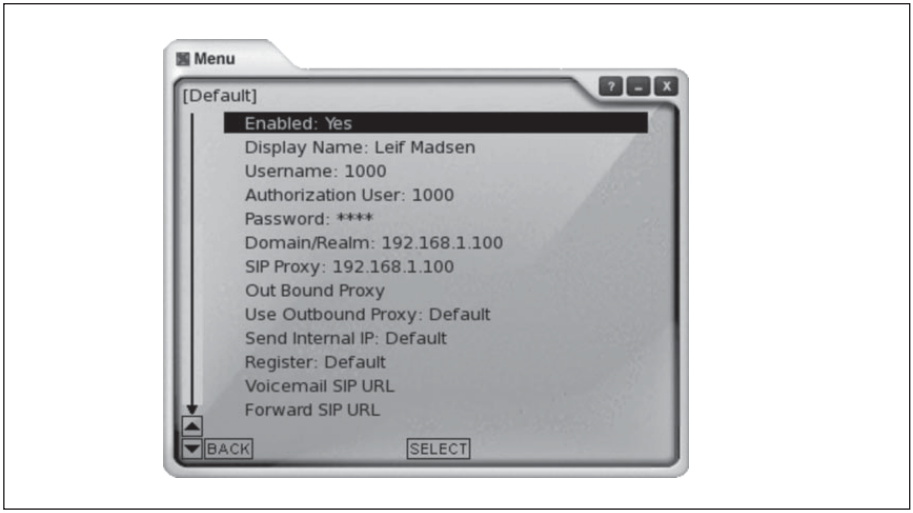



***Рис. 4.3.*** *Пользовательская конфигурация X-Lite*

CLI Asterisk будет предоставлен следующим образом:

```
*CLI>
```

Если Asterisk уже была запущена до изменения sip.conf, как рекомендуется в данной главе, перезагрузите диалплан и SIP-канал с помощью следующих двух команд:

```
*CLI> dialplan reload

*CLI> sip reload
```

В клиенте программного телефона X-Lite закройте окно Settings (Настройки), щелкая по кнопке BACK (НАЗАД) до тех пор, пока не будут закрыты все окна. Вы должны увидеть, что X-Lite пытается зарегистрироваться в Asterisk, и в случае успеха в CLI Asterisk будет выведено следующее:

```
-- Registered SIP '1000' at 192.168.1.250 port 5061 expires 3600
```

Проверить статус регистрации можно в любой момент следующим образом:

```
*CLI> sip show peers

Name/username Host            Dyn  Nat ACL Port  Status
1000/1000     192.168.1.250   D    N       5061  OK (63 ms)
1 sip peers [1 online , 0 offline]
```

Более подробная информация, представленная ниже, может быть получена с помощью команды `sip show peer 1000`:

```
*CLI> sip show peer 1000


  * Name           : 1000
  Secret           : <Not set>
  MD5Secret        : <Not set>
  Context          : phones
  Subscr.Cont.     : <Not set>
  Language         :
  AMA flags        : Unknown
  Transfer mode    : open
  CallingPres      : Presentation Allowed, Not Screened
  Callgroup        :
  Pickupgroup      :
  Mailbox          :
  VM Extension     : asterisk
  LastMsgsSent     : 32767/65535
  Call limit       : 0
  Dynamic          : Yes
  Callerid         : "" <>
  MaxCallBR        : 384 kbps
  Expire           : 1032
  Insecure         : no
  Nat              : RFC3581
  ACL              : No
  T38 pt UDPTL     : No
  CanReinvite      : Yes
  PromiscRedir     : No
  User=Phone       : No
  Video Support    : No
  Trust RPID       : No
```

```
Send RPID        : No
Subscriptions    : Yes
Overlap dial     : Yes
DTMFmode         : rfc2833
LastMsg          : 0
ToHost           :
Addr->IP         : 192.168.1.250 Port 5061
Defaddr->IP      : 0.0.0.0 Port 5060
Def. Username    : 1000
SIP Options      : (none)
Codecs           : 0x8000e (gsm|ulaw|alaw|h263)
Codec Order      : (none)
Auto-Framing     : No
Status           : Unmonitored
Useragent        : X-Lite release 1105d
Reg. Contact     : sip:1000@192.168.1.250:5061
```

## Телефонные аппараты Polycom IP 430

Многие считают, что конфигурировать телефоны Polycom сложно. Насколько нам известно, такие выводы делаются по двум причинам: 1) ужасный веб-интерфейс Polycom, или 2) тяжелый и запутанный процесс автоматической подготовки к работе.

Что касается пункта 1, мы согласны. Веб-интерфейс телефонов Polycom является одним из самых неприятных веб-интерфейсов из всех, когда-либо созданных для IP-телефонов. Мы его не используем и вам не советуем[1].

Итак, нам остается только кое-какая конфигурация с использованием сервера. К счастью, в этом отношении IP-телефоны Polycom просто великолепны, настолько, что им даже можно простить веб-интерфейс. Конфигурации телефонов хранятся в файлах на сервере. Каждый телефон находит сервер, загружает с него соответствующие файлы и применяет их к себе.

### DHCP-сервер

Если вы не можете управлять используемым DHCP-сервером, вам придется вручную вводить в телефон информацию о FTP-сервере. Для этого необходимо перезагрузить телефон, нажать кнопку setup (настройка) до того, как телефон начнет процесс загрузки, и задать адрес FTP-сервера в небольшом меню загрузки, предлагаемом этим типом телефонов.

### Протокол, используемый для загрузки

Телефоны Polycom могут загружать свою конфигурацию по одному из трех протоколов: TFTP, HTTP и FTP.

---

[1] Как ни странно, но он все-таки выполняет одну полезную функцию – обеспечивает возможность входить в телефон через броузер и запрашивать его конфигурацию.

Сразу же хотим попросить избегать TFTP. Он не обеспечивает необходимой безопасности, и телефон не может использовать информацию о дате для определения того, какие версии файлов являются самыми последними. Этот протокол работает, но есть лучшие способы, и мы здесь не собираемся обсуждать TFTP.

Телефоны Polycom могут извлекать свои данные конфигурации и с помощью HTTP, но он не получил широкого распространения, и мы также не собираемся останавливаться на нем.

В настоящее время FTP является предпочтительным методом получения телефонами Polycom своей конфигурации. Он прекрасно работает, его довольно просто конфигурировать, и он получил хорошую поддержку сообщества.

## FTP

В настоящее время FTP – наш любимый способ конфигурации телефонов Polycom. В системе CentOS при выполнении следующей команды будет установлен VSFTPD, Very Secure FTP Daemon:

```
# yum -y install vsftpd
```

Затем для обеспечения защиты необходимо предотвратить анонимные входы в систему. Для этого вносим простое изменение в конфигурационный файл vsftpd, находящийся в папке /etc/vsftpd/vsftpd.conf:

```
# anonymous_enable=NO
```

Перезапустите сервер с помощью команды `service vsftpd restart`. Чтобы гарантировать запуск демона после каждой перезагрузки, выполняем команду `chkconfig vsftpd on`.

Теперь необходимо создать пользовательскую учетную запись и группу, которые будут использоваться для телефонных аппаратов Polycom:

```
# groupadd PlcmSpIp
# useradd PlcmSpIp -g PlcmSpIp -p PlcmSpIp
# passwd PlcmSpIp
```

Задаем пароль PlcmSpIp (стандартный пароль FTP для аппаратов Polycom). Его можно изменить, но после этого придется вручную настраивать каждый аппарат, чтобы сообщить ему его нестандартные учетные данные[1].

Для большей безопасности убедимся, что FTP-сервер хранит эту учетную запись в «темнице» `chroot`:

---

[1] Можно придумывать для телефонов невероятно сложные пароли, которые невозможно угадать, но, если вы не собираетесь вводить их в каждый телефон вручную, их имя пользователя и пароль на FTP-сервере придется передавать с DHCP-сервера. Любое устройство, которое может регистрироваться в сети телефонной связи, способно получать информацию с DHCP-сервера. Мы не предлагаем игнорировать безопасность, просто не думайте, что создание индивидуального пароля для каждого телефона улучшит ее.

```
# echo PlcmSpIp >> /etc/vsftpd/vsftpd.chroot_list
```

Вот примерно то, что должно быть сделано, чтобы подготовить операционную систему к предоставлению необходимых сервисов телефонам.

## Конфигурационные файлы Polycom

Хотя кажется, что разных файлов, которые необходимы для подготовки к работе телефона Polycom, так много, разобраться с ними довольно просто.

**bootROM.** Лучше всего охарактеризовать этот файл как BIOS и операционную систему телефона. Вероятно, можно дать и более формальное определение, но зачем усложнять? bootROM не нужно регулярно обновлять, но лучше отслеживать выпускаемые версии на тот случай, если в более новом bootROM появятся полезные в вашей среде возможности. Этот файл будет называться bootrom.ld.

**Образ приложения.** Поскольку телефоны Polycom могут поддерживать другие протоколы VoIP (например, поддерживается MGCP (Media Gateway Control Protocol – протокол контроля медиа-шлюзов)), протокол, который будет использовать данный телефон, является частью образа приложения, загружаемого и выполняемого телефоном. Если в телефоне уже имеется соответствующий образ, этот файл на FTP-сервере фактически не нужен; однако обычно он доступен, чтобы обеспечить телефонам возможность получить последнюю версию протокола. К вам могут попасть телефоны, использующие не последнюю версию, поэтому наличие образа с текущей версией протокола гарантирует, что все телефоны будут отвечать самым последним требованиям.

**Файл sip.cfg file.** Обычно в системе имеется только одна версия этого файла, но ему может быть присвоено любое имя, поэтому его версий может быть столько, сколько потребуется. Например, если в организации в ходу два языка, некоторые пользователи могут предпочесть использовать в своих телефонах французский, другие – английский. В этом случае для каждого варианта пришлось бы создать два файла: french.sip.conf и english.sip.conf. Этот файл можно назвать как угодно, но лучше выбирать имя, имеющее смысл, чтобы в будущем администраторы могли понять, почему были приняты именно такие проектные решения.

**Основной конфигурационный файл для каждого телефона.** Это очень простой и небольшой файл. Его имя соответствует MAC-адресу телефона (то есть у каждого телефона должна быть собственная копия этого файла). Из этого файла телефон получает информацию о том, какие еще файлы необходимо загрузить, чтобы выполнить свою конфигурацию. Этот конфигурационный файл все телефоны читают первым. В нем находится ссылка на образ приложения, который будет использовать данный телефон (в настоящее время названный sip.ld), а также имена XML-файлов, содержащих параметры для этого телефона (файлы .cfg). Основной конфигурационный файл телефона может выглядеть так:

```
'<?xml version="1.0" standalone="yes"?>'
'<!-- Default Master SIP Configuration File-->'
'<!-- Edit and rename this file to <Ethernet-address>.cfg for each phone.-->'
'<!-- $Revision: 1.14 $ $Date: 2005/07/27 18:43:30 $ -->'
'<APPLICATION APP_FILE_PATH="sip.ld"
  CONFIG_FILES="phone1.cfg, sip.cfg"
  MISC_FILES=""
  LOG_FILE_DIRECTORY=""
  OVERRIDES_DIRECTORY=""
  CONTACTS_DIRECTORY=""
/>'
```

Обратите внимание на имя файла приложения, которое должен будет использовать данный телефон, и конфигурационные файлы, которые он будет пытаться найти и применить.

**Специальный конфигурационный файл телефона.** Мы рекомендуем присваивать файлам phone1.cfg имена, имеющие смысл. Например, SET<xxx>.cfg (как SET201.cfg), соответственно добавочному номеру телефона, или FLOOR4CUBE23.cfg, или, может быть, BOB_SMITHS_IP430_SET.cfg, или что угодно, что вам больше нравится. Как лучше всего называть их? Ответим вопросом на вопрос. Скажем, имеется 100 таких телефонов. При просмотре содержимого папки /home/PlcmSpIp как вы хотели бы, чтобы выглядели эти 100 конфигурационных файлов для телефонов?

**Сбои.** Настройки, заданные непосредственно в телефоне, будут храниться в файловой системе телефона и могут иметь приоритет перед параметрами конфигурационных файлов. Если возникают проблемы с внесением изменений в телефон, попытайтесь переформатировать его. Это заставит его принять параметры, содержащиеся в конфигурационных файлах.

## Телефон Cisco 7960

Почтенный старец C7960 – сейчас уже часть истории VoIP. Это один из первых SIP-телефонов, к которому действительно можно было относиться серьезно. Единственное, на что можно пожаловаться, – его цена. Это «Cadillac» среди SIP-телефонов (в том смысле, что у них есть всякие прибамбасы, но это не оправдывает их цены и иногда они немного старомодны).

Если вы можете достать один из них, вы получите превосходный SIP-телефон. Если покупаете новый, будьте готовы выложить кругленькую сумму.

Один из аспектов несоответствия этого телефона духу времени – возможность удаленной подготовки к работе только посредством TFTP. TFTP потерял доверие сетевых специалистов из-за отсутствия поддержки аутентификации и шифрования, но поскольку это единственный способ удаленно настроить данный телефон, нам придется использо-

вать демон tftp-сервера. Установить tftp-сервер можно с помощью следующей команды:

```
# yum install -y tftp-server
```

После установки TFTP-сервер необходимо активировать. Для этого в файле /etc/xinetd.d/tftp строку `disable=yes` меняем на `disable=no`.

```
service tftp
{
   socket_type     = dgram
   protocol        = udp
   wait            = yes
   user            = root
   server          = /usr/sbin/in.tftpd
   server_args     = -s /tftpboot
   disable         = no
   per_source      = 11
   cps             = 100 2
   flags           = IPv4
}
```

Затем запускаем TFTP-сервер, выполнив команду

```
# service xinetd restart
```

Проверить, выполняется ли сервер, можно с помощью следующей команды:

```
# chkconfig --list | grep tftp
  tftp: on
```

Поскольку возвращено `tftp: on`, сервер активирован и выполняется.

Телефоны Cisco по умолчанию загружаются с собственным протоколом связи, SCCP (или Skinny). Мы покажем, как конфигурировать телефон, но из-за узкоспециализированности Cisco и ее телефонов вам понадобится получить встроенные программы SIP у своего провайдера услуг связи. Также для Asterisk существует две реализации: модули chan_sccp и chan_skinny, – но их рассмотрение выходит за рамки данной книги.

Мы зарегистрируем наш телефон Cisco как SIP-клиента для станции, которую мы сконфигурировали в разделе «Конфигурация оборудования Zaptel». Следующий конфигурационный файл следует сохранить под именем SIP<mac>.cnf, где <mac> представляет MAC-адрес конфигурируемого телефонного устройства. Поместите этот файл в папку /tftpboot/ своего сервера:

```
# Конфигурация линии 1
line1_name: "1000"
line1_authname: "1000"
line1_shortname: "Jimmy Carter"
line1_password: ""
line1_displayname: ""
```

```
# Имя телефона, которое отображается в верхнем правом углу дисплея телефона
phone_label: "aristotle" ; Не оказывает влияния на обмен сообщениями по протоколу SIP

# Пароль, используемый для доступа к телефону через консоль или по протоколу
telnet, не более 31 символа
phone_password: "cisco"
```

Затем в файле SIPDefault.cnf, также находящемся в папке /tftpboot/ сервера, задаем адрес для регистрации. `proxy1_address` (адрес прокси1) будет содержать IP-адрес вашего сервера Asterisk, на котором телефон должен зарегистрироваться для линии 1. Параметр `image_version` (версия образа) указывает версию файлов .loads и .sb2, загружаемую телефоном в память.

```
image_version: P0S3-08-4-00
proxy1_address: 192.168.1.100
```

Нам нужен один дополнительный файл, OS79XX.TXT. В нем есть лишь одна строка – версия файлов .bin и .sbn, которая должна быть загружена в память:

```
P003-08-4-00
```

Чтобы наш Cisco 7960 использовал эти файлы, мы должны указать ему, откуда он может взять свою конфигурацию. Если используется DHCP-сервер вашего Linux-сервера, сделать это можно, добавив в файл /etc/dhcpd.conf строку

```
option tftp-server-name "192.168.1.100";
```

которая содержит IP-адрес сервера, являющегося хостом для TFTP-сервера (предполагается, конечно, что TFTP-сервер сконфигурирован по этому адресу. Этот адрес использовался для нашего Asterisk-сервера, и опять мы предполагаем, что TFTP-сервер установлен на одном сервере с Asterisk). Подробнее о конфигурации DHCP-сервера рассказывалось в разделе «DHCP-сервер»:

```
ddns-update-style interim;
ignore client-updates;

subnet 192.168.1.0 netmask 255.255.255.0 {
     option routers 192.168.1.1;
     option subnet-mask 255.255.255.0;
     option domain-name-servers 192.168.1.1;
     option tftp-server-name "192.168.1.100";
     option ntp-servers pool.ntp.org;
     option time-offset -18000;

    range dynamic-bootp 192.168.1.128 192.168.1.254;
    default-lease-time 21600;
    max-lease-time 43200;
}
```

В качестве альтернативы можно выполнить настройку с самого телефона и вручную задать использование альтернативного, а не предоставляемого DHCP-сервером, TFTP-сервера. Для этого нажмите кнопку settings (настройки) (на моделях G телефонов Cisco она выглядит как квадратик с галочкой внутри; G означает Global). Затем понадобится разблокировать настройки, нажав кнопку 9. Пароль по умолчанию – cisco. Как только телефон будет разблокирован, нажмите кнопку 3 номеронабирателя, чтобы войти в раздел Network Configuration (Настройки сети). С помощью прокрутки перейдите вниз к опции 32, Alternate TFTP (Альтернативный TFTP), и задайте для нее значение YES. Затем поднимитесь к опции 7 и введите IP-адрес TFTP-сервера, с которого вы хотите загружаться. Подтвердите настройки и выйдите из меню, пока телефон перезагружается. Также, используя комбинацию кнопок *+6+settings, можно перезагружать телефон в любое время.

Понаблюдать за тем, как телефон извлекает конфигурацию с TFTP-сервера, можно с помощью команды `tshark` (`yum install ethereal`). Задайте фильтрацию по порту 69 следующим образом:

```
# tshark port 69
```

Это позволит просматривать сетевой трафик телефона, запрашивающего данные с TFTP-сервера.

Если все пройдет успешно, вы увидите, что ваш телефон зарегистрировался в Asterisk!

## Linksys SPA–942

С момента приобретения Sipura Technologies компания Linksys выпускает недорогие VoIP-телефоны и аналоговые терминальные адаптеры (ATA). Linksys многое позаимствовала у Cisco. Почитайте книгу Клейтона М. Кристенсена (Clayton M. Christensen) «The Innovator's Dilemma» (издательство HarperCollins), и стратегия Cisco относительно Linksys станет для вас яснее.

Продукты Linksys (и Sipura) снискали славу за свое превосходное качество, особенно с учетом цены, но они известны также тем, что их ужасно сложно настраивать.

Это, главным образом, объясняется тем, что их GUI конфигурации предлагает сотни настраиваемых параметров.

Но нас это не волнует. Вот то, что необходимо знать, чтобы настроить SPA-942 (и, надеемся, большинство VoIP-устройств Linksys) для работы с системой Asterisk.

### Как выполнить вход в телефон

Прежде всего необходимо получить IP-адрес телефона, чтобы можно было войти в его GUI. На самом телефоне найдите кнопку со значком,


***Рис. 4.4.*** *Клавиатура SPA-942*

который выглядит как листок бумаги с загнутым углом (сразу под кнопкой с конвертом). Это кнопка Settings (Настройки) – рис. 4.4.

Чтобы получить IP-адрес телефона, нажмите кнопку Settings (Настройки), а затем – кнопку 9 (или используйте кнопку со стрелками и с помощью прокрутки перейдите к пункту Network (Сеть)). После этого нажмите кнопку select (выбрать) (под жидкокристаллическим дисплеем располагается ряд из 4 кнопок, select – крайняя слева). Во втором поле должен быть указан IP-адрес телефона.

Теперь откройте броузер, введите IP-адрес в адресную строку, нажмите кнопку Enter (Ввод) – и вы увидите окно Info (Информация) телефона.

## Регистрация телефона в Asterisk

В верхнем правом углу экрана выберите ссылку Admin Login (Вход под учетной записью администратора). При этом появится несколько новых вкладок, таких как Regional (Региональные), Phone (Телефон), Ext 1, Ext 2 и User (Пользователь).

Выберите вкладку Ext 1, с помощью которой мы выполним настройку первой линии. Выберите в меню следующие опции:

1. General ⇒ Line Enable ⇒ yes (Общие ⇒ Линия доступна ⇒ да).
2. NAT Settings ⇒ NAT Mapping Enable ⇒ no (Настройки NAT ⇒ Отображение NAT доступно ⇒ нет).
3. NAT Settings ⇒ NAT Keep Alive Enable ⇒ no (Настройки NAT ⇒ Поддержка NAT доступна ⇒ нет).
4. Proxy and Registration ⇒ Proxy ⇒ [Введите IP-адрес Asterisk (например, 192.168.1.100)] (Прокси и регистрация ⇒ Прокси ⇒ [Введите IP-адрес Asterisk]).
5. Proxy and Registration ⇒ Register ⇒ yes (Прокси и регистрация ⇒ Регистрация ⇒ да).

6. Proxy and Registration ⇒ Make Call Without Reg ⇒ no (Прокси и регистрация ⇒ Звонить без регистрации ⇒ нет).
7. Proxy and Registration ⇒ Ans Call Without Reg ⇒ no (Прокси и регистрация ⇒ Отвечать на звонок без регистрации ⇒ нет).
8. Subscriber Information ⇒ Display Name ⇒ Caller ID information (Информация об абоненте ⇒ Отображать имя ⇒ Информация об ID звонящего).
9. Subscriber Information ⇒ User ID ⇒ 1000 (Информация об абоненте ⇒ ID пользователя ⇒ 1000).
10. Subscriber Information ⇒ Password ⇒ [Оставьте незаполненным, если используете простую конфигурацию, описанную ранее в этой главе] (Информация об абоненте ⇒ Пароль ⇒ [Оставьте незаполненным, если используете простую конфигурацию, описанную ранее в этой главе]).
11. Subscriber Information ⇒ Use Auth ID ⇒ yes (Информация об абоненте ⇒ Использовать ID авторизации ⇒ да).
12. Subscriber Information ⇒ Auth ID ⇒ 1000 (Информация об абоненте ⇒ ID авторизации ⇒ 1000).
13. Audio Configuration ⇒ Preferred Codec ⇒ G711u (Конфигурация аудио ⇒ Предпочтительный кодек ⇒ G711u).
14. Audio Configuration ⇒ Use Pref Codec Only ⇒ no (Конфигурация аудио ⇒ Использовать только предпочтительный кодек ⇒ нет).
15. Audio Configuration ⇒ Silence Supp Enable ⇒ no (Конфигурация аудио ⇒ Поддержка тишины доступна ⇒ нет).
16. Audio Configuration ⇒ DTMF Tx Method ⇒ Auto (Конфигурация аудио ⇒ Метод DTMF Tx ⇒ Автоматически).
17. Подтвердите все изменения

И это все! Теперь ваш телефон должен быть зарегистрирован в Asterisk. Вы узнаете об этом по кнопке-индикатору, расположенному рядом с жидкокристаллическим дисплеем, – он изменит цвет с оранжевого на зеленый.

## Конфигурация диалплана для выполнения тестирования

Чтобы ваш телефон мог звонить на другие телефоны (или, для многоканального телефона, звонить самому себе), необходимо внести изменения в файл extensions.conf. Взяв за основу то, что было сделано в разделе «Настройка диалплана для выполнения тестовых вызовов», добавим следующие строки в контекст `[internal]`:

```
exten => 1000,1,Verbose(1|Extension 1000)
exten => 1000,n,Dial(SIP/1000,30)
exten => 1000,n,Hangup()
```

Если имеется два телефона или сконфигурировано несколько линий, можно продублировать предыдущие настройки, заменив 1000 на другой добавочный номер.

# Подключение к поставщику сервисов SIP

С появлением интернет-телефонии по всему миру возникло множество телефонных компаний, использующих интернет-технологии! Поэтому выбор у нас огромен. Многие из этих поставщиков сервисов обеспечивают возможность подключения частной Asterisk-системы к их сетям[1], а некоторые даже используют Asterisk сами!

Следующая конфигурация должна обеспечить подключение к поставщику сервисов интернет-телефонии (Internet Telephony Service Provider, ITSP)[2], хотя невозможно предугадать, какую конфигурацию потребует конкретный поставщик. В идеале поставщик сам предоставит вам настройки, которые необходимы для подключения вашей системы. Однако не все поддерживают Asterisk, поэтому мы собираемся предложить универсальную конфигурацию, на выполнение которой уйдет лишь несколько минут и которая должна помочь вам:

```
[мой_поставщик_услуг]
type=peer
host=10.251.55.100
fromuser=мой_уникальный_id
secret=мой_секретный пароль
context=incoming_calls
dtmfmode=rfc2833
disallow=all
allow=gsm
allow=ulaw
```

### Конфигурация локального межсетевого экрана

Если межсетевой экран ip-таблиц используется на одном компьютере с сервером Asterisk, выполнив следующие команды, можно открыть порт 5060 для обмена сигналами по протоколу SIP и порты от 10000 до 20000 для RTP-трафика. Также диапазон RTP-портов можно сузить в файле rtp.conf, находящемся в папке /etc/asterisk. Замечательная книга по организации межсетевых экранов с помощью ip-таблиц – «Linux Firewalls» (издательство Novell Press) Стива Суэринга (Steve Suehring) и Роберта Циглера (Robert Ziegler).

```
# iptables -I RH-Firewall-1-INPUT -p udp --dport 5060 -j ACCEPT
# iptables -I RH-Firewall-1-INPUT -p udp --dport 10000:20000 -j ACCEPT
# service iptables save
```

Помните, что это откроет порты 5060 и от 10000 до 20000 для всего UDP-трафика из любого источника.

---

[1] Обязательно проверяйте политику провайдера, к которому вы планируете подключиться, поскольку некоторые из них могут запрещать использование офисной АТС с их сервисом.

[2] Также их называют провайдерами VoIP (VoIP Service Provider, VSP).

```
deny=0.0.0.0/0
permit=10.251.55.100/32
insecure=invite
```

Большинство приведенных настроек должны быть вам знакомы, но если нет, далее дается их краткое описание.

Задавая тип `peer`, мы указываем Asterisk, что при получении сообщения `INVITE` (Приглашение) (когда поставщик присылает вызов) нужно сравнивать не имя [`мой поставщик сервисов`], а IP-адрес, указанный в этом сообщении. Параметр `host` – это IP-адрес, на который мы будем направлять наши вызовы, и этот IP-адрес будет сопоставляться при получении вызова от поставщика.

Параметр `fromuser` (от пользователя) влияет на то, как структурировано наше сообщение `INVITE` при отправке вызова поставщику сервисов.

### Сопоставление имени пользователя, а не IP-адреса

Некоторые поставщики услуг для отправки своих вызовов могут использовать вместо протокола Session Initiation Protocol множество IP-адресов, требуя от вас создания отдельной учетной записи типа `peer` для каждого IP-адреса. Если известны не все IP-адреса, вероятно, придется сравнивать имена пользователей. В этом случае потребуется лишь немного изменить формат описания поставщика сервисов. Самое большое изменение, на которое следует обратить внимание, – то, что вам понадобится задать [*`заголовок_поставщика_услуг`*] как имя пользователя, которому ваш поставщик сервисов собирается направлять вызовы. Также мы изменили тип `peer` (равноправный) на `friend` (дружественный), что с точки зрения Asterisk создает типы и `user` (пользователь), и `peer`, где тип `user` будет сравниваться раньше `peer`:

```
[мой_уникальный_id]
type=friend
host=10.251.55.100
fromuser=мой_уникальный_id
secret=мой_секретный пароль
context=incoming_calls
dtmfmode=rfc2833
disallow=all
allow=gsm
allow=ulaw
insecure=invite
```

Обратите внимание, что удалены параметры `deny` (отклонить) и `permit` (разрешить), поскольку IP-адреса, с которых будут поступать вызовы, могут быть неизвестны. Если вдруг эти адреса известны и вы по-прежнему хотите сравнивать их, параметры `deny` и `permit` для IP-адресов можно восстановить.

Если имя пользователя задано в параметре `fromuser`, при отправке вызова поставщику меняются поля `From:` (От:) и `Contact:` (Контакт:) сообщения `INVITE`. Этого может требовать сам поставщик, если он использует эти поля в процедуре аутентификации. То, где Asterisk меняет заголовок, можно увидеть в следующих двух фрагментах кода.

Без параметра `fromuser`:

```
Audio is at 66.135.99.122 port 18154
Adding codec 0x2 (gsm) to SDP
Adding codec 0x4 (ulaw) to SDP
Adding non-codec 0x1 (telephone-event) to SDP
Reliably Transmitting (no NAT) to 10.251.55.100:5060:
INVITE sip:15195915119@10.251.55.100 SIP/2.0
Via: SIP/2.0/UDP 66.135.99.122:5060;branch=z9hG4bK32469d35;rport
From: "asterisk" <sip:asterisk@66.135.99.122>;tag=as4975f3ff
To: <sip:15195915119@10.251.55.100>
Contact: <sip:asterisk@66.135.99.122>
Call-ID: 58e3dfb2584930cd77fe989c00986584@66.135.99.122
CSeq: 102 INVITE
User-Agent: Asterisk PBX
Max-Forwards: 70
Date: Fri, 20 Apr 2007 14:59:24 GMT
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY
Supported: replaces
Content-Type: application/sdp
Content-Length: 265
```

С параметром `fromuser`:

```
Audio is at 66.135.99.122 port 11700
Adding codec 0x2 (gsm) to SDP
Adding codec 0x4 (ulaw) to SDP
Adding non-codec 0x1 (telephone-event) to SDP
Reliably Transmitting (no NAT) to 10.251.55.100:5060:
INVITE sip:15195915119@10.251.55.100 SIP/2.0
Via: SIP/2.0/UDP 66.135.99.122:5060;branch=z9hG4bK635b0b1b;rport
From: "asterisk" <sip:мой_уникальный_id@66.135.99.122>;tag=as3186c1ba
To: <sip:15195915119@10.251.55.100>
Contact: <sip:мой_уникальный_id@66.135.99.122>
Call-ID: 0c7ad6156f92e70b1fecde903550a12f@66.135.99.122
CSeq: 102 INVITE
User-Agent: Asterisk PBX
Max-Forwards: 70
Date: Fri, 20 Apr 2007 15:00:30 GMT
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY
Supported: replaces
Content-Type: application/sdp
Content-Length: 265
```

Параметры `deny` и `permit` используются для отклонения всех входящих вызовов к этому участнику сети от всех IP-адресов, кроме заданного параметром `permit`. Это просто мера безопасности, которая обеспечивает поступление к данному участнику сети трафика только с ожидаемого IP-адреса.

В конце видим выражение `insecure=invite`, которое может быть необходимо для вашего поставщика, потому что источник сообщения `INVITE` может находиться на платформе сервера, но передаваться через его прокси-сервер SIP. По сути, это означает, что в строке `Contact` (Контакт) (поле сообщения `INVITE`, которое вы проверяете при приеме вызова от своего поставщика) может быть совсем не тот IP-адрес, по которому находится участник сети. Это выражение указывает Asterisk игнорировать данное несоответствие и принимать приглашение в любом случае.

Возможно, понадобится задать и `invite=invite,port`, если адрес порта также не соответствует тому, что ожидает Asterisk.

Теперь в разделе `[general]` файла sip.conf требуется задать один дополнительный параметр: `register`. `register` (реестр) укажет поставщику сервисов, куда направлять адресованные нам вызовы. Так Asterisk говорит поставщику сервисов: «Эй! Если ты получил вызов ко мне, пошли его на IP-адрес 10.251.55.100». Параметр `register` имеет следующий вид:

```
register => имяпользователя:секрет@мой.поставщик_сервисов.tld
```

Теперь осталось только сконфигурировать простой диалплан для обработки входящих вызовов и отправки вызовов через поставщика сервисов. Мы собираемся внести коррективы в диалплан, который начали создавать в разделе «Настройка диалплана для выполнения тестовых вызовов» данной главы. Строки, выделенные курсивом, – это новые части, добавленные в диалплан. Все остальное взято без изменений из предыдущего раздела[1].

```
[globals]

[general]

[default]
exten => s,1,Verbose(1|Unrouted call handler)
exten => s,n,Answer()
exten => s,n,Wait(1)
exten => s,n,Playback(tt-weasels)
exten => s,n,Hangup()

[incoming_calls]
exten => _X.,1,NoOp()
exten => _X.,n,Dial(SIP/1000)

[outgoing_calls]
exten => _X.,1,NoOp()
exten => _X.,n,Dial(SIP/мой_поставщик_сервисов/${EXTEN})
```

[1] Также предполагается, что сконфигурирован по крайней мере один добавочный номер SIP из предыдущего раздела.

```
[internal]
exten => 1000,1,Verbose(1|Extension 1000)
exten => 1000,n,Dial(SIP/1000,30)
exten => 1000,n,Hangup()

exten => 500,1,Verbose(1|Echo test application)
exten => 500,n,Echo()
exten => 500,n,Hangup()

[phones]
include => internal
include => outgoing_calls
```

# Соединение двух серверов Asterisk по протоколу SIP

Может настать время, когда у вас появится два сервера Asterisk и вам захочется передавать вызовы между ними. К счастью, здесь нет особых сложностей, хотя имеются некоторые странности, с которыми придется справляться, но с точки зрения конфигурации на самом деле это совсем не так уж трудно.

### Конфигурация локального межсетевого экрана

Если ip-таблицы используются на одном компьютере с сервером Asterisk, с помощью следующих команд можно открыть порт 5060 для обмена сигналами по протоколу SIP и порты от 10000 до 20000 для RTP-трафика. Также диапазон RTP-портов можно сузить в файле rtp.conf, находящемся в папке /etc/asterisk. Замечательная книга по межсетевым экранам для ip-таблиц – «Linux Firewalls» (издательство Novell Press) Стива Суэринга (Steve Suehring) и Роберта Циглера (Robert Ziegler).

```
# iptables -I RH-Firewall-1-INPUT -p udp --dport 5060 -j ACCEPT
# iptables -I RH-Firewall-1-INPUT -p udp --dport 10000:20000 -j ACCEPT
# service iptables save
```

Помните, что это откроет порты 5060 и от 10000 до 20000 для всего UDP-трафика из любого источника.

Наша топология будет состоять из SIP-телефона (Элис (Alice)), зарегистрированного в Asterisk A (Торонто (Toronto)), и SIP-телефона (Боб (Bob)), зарегистрированного в Asterisk B (Осака (Osaka)). К концу данного раздела вы сможете с помощью пары серверов Asterisk производить звонки от Элис к Бобу (и наоборот) – рис. 4.5. Это типовой сценарий, когда имеется два физических местоположения, например ком-

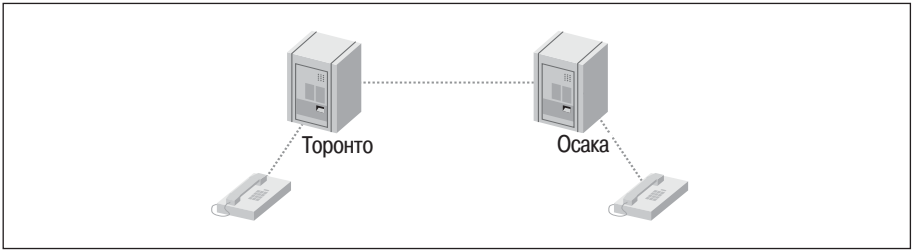



***Рис. 4.5.*** *Топология объединения каналов SIP*

пания с несколькими офисами, и требуется обеспечить единую логическую топологию расширения.

Прежде всего давайте сконфигурируем серверы Asterisk.

## Конфигурация серверов Asterisk

У нас есть пара серверов Asterisk, назовем их Торонто и Осака, и мы собираемся зарегистрировать их друг на друге. В этом сценарии будем использовать самый элементарный файл sip.conf. Как и в случае с рассматриваемой ранее в данной главе конфигурацией SIP-телефона, это не лучший способ, но он работает.

Вот конфигурация сервера Торонто:

```
[general]
register => toronto:welcome@192.168.1.101/osaka

[osaka]
type=friend
secret=welcome
context=osaka_incoming
host=dynamic
disallow=all
allow=ulaw
```

И конфигурация сервера Осака:

```
[general]
register => osaka:welcome@192.168.2.202/toronto

[toronto]
type=friend
secret=welcome
context=toronto_incoming
host=dynamic
disallow=all
allow=ulaw
```

Многие из приведенных опций могут быть вам знакомы, но давайте на всякий случай остановимся на них подробнее.

Вторая строка файла указывает серверу Asterisk зарегистрироваться на другом сервере. Таким образом мы сообщаем удаленному серверу

Asterisk, куда направлять вызовы, когда он пожелает обратиться к нашему локальному серверу Asterisk. Помните, мы предупреждали о небольших странностях в конфигурации? Видите в конце строки регистрации слэш и имя удаленного сервера Asterisk? Так удаленный сервер Asterisk получает информацию о том, какое краткое имя использовать при вызове. Если не добавить этого, при попытке дальнего конца сделать вызов в окне командной строки Asterisk появится следующее сообщение:

```
[Apr 22 18:52:32] WARNING[23631]: chan_sip.c:8117 check_auth: username
                                  mismatch, have <toronto>, digest has <s>
```

Таким образом, добавляя слэш и имя, мы сообщаем противоположному концу, что должно быть указано в качестве краткого имени пользователя в поле `Proxy Authorization` (Авторизация прокси) SIP-сообщения `INVITE`.

Весь остальной файл занимает блок авторизации, используемый для управления входящими и исходящими вызовами другого сервера Asterisk. Сервер Торонто использует блок авторизации `[osaka]`, и сервер Осака использует блок `[toronto]`. Определен тип `friend`, что позволяет принимать и направлять вызовы к другому серверу Asterisk. Параметр `secret` (секрет) – это пароль, который должна использовать другая система при аутентификации. Параметр `context` (контекст) указывает, в какой части диалплана (extensions.conf) обрабатываются входящие вызовы. Для параметра `host` задано значение `dynamic` (динамический), это указывает серверу Asterisk на то, что противоположный конец сообщит свой IP-адрес, на который следует направлять адресованные ему звонки, при регистрации. Наконец, с помощью параметров `disallow` (запретить) и `allow` (разрешить) можно определять, какие кодеки будут использоваться при общении с противоположным концом.

После сохранения файла и перезагрузки SIP-канала на обоих серверах Asterisk (выполнение команды `sip reload` из консоли Asterisk) в окне командной строки должно быть выведено примерно следующее сообщение, что свидетельствует об успешной регистрации удаленного сервера:

```
*CLI> -- Saved useragent "Asterisk PBX" for peer toronto
```

(Для равноправного участника сети торонто сохранен агент пользователя "офисная АТС Asterisk")

После выполнения команды `sip show peers` статус хоста `Unspecified` (Не определен) должен быть замен IP-адресом удаленного сервера:

```
*CLI> sip show peers
Name/username Host            Dyn Nat ACL Port  Status
toronto/osaka 192.168.2.202   D           5060  Unmonitored
```

Убедиться в успешности собственной регистрации можно, выполнив команду `sip show registry` из консоли Asterisk:

```
*CLI> sip show registry
Host                Username  Refresh State    Reg.Time
192.168.1.101:5060 osaka      105 Registered   Sun, 22 Apr 2007 19:13:20
```

Теперь, когда оба сервера Asterisk довольны друг другом, займемся конфигурацией пары SIP-телефонов, чтобы иметь возможность позвонить.

## Конфигурация SIP-телефона

Подробнее о конфигурации SIP-телефонов в Asterisk рассказывается в разделе «Конфигурация канала FXS для аналогового телефона» данной главы. Ниже представлена конфигурация SIP-телефона в файле sip.conf для каждого из двух серверов, которые будут использоваться в диалплане в следующем разделе, предоставляющая нам две конечные точки для установления соединения. Эти строки должны быть добавлены в конце файла sip.conf для каждого соответствующего сервера.

Файл sip.conf для сервера Торонто:

```
[1000]
type=friend
host=dynamic
context=phones
```

Файл sip.conf для сервера Осака:

```
[1001]
type=friend
host=dynamic
context=phones
```

Теперь для сервера Торонто должен быть зарегистрирован добавочный номер 1000, а для сервера Осака – 1001. Убедиться в этом можно с помощью команды `sip show peers`. Далее мы собираемся сконфигурировать логику диплплана, что позволит производить звонки с одного добавочного номера на другой.

## Конфигурация диалплана

Теперь можно сконфигурировать простой диалплан для каждого сервера, который позволит выполнять звонки между зарегистрированными телефонами: один для Торонто, а другой – для Осаки. В разделе «Работа с конфигурационными файлами интерфейсов» данной главы мы создали простой файл extensions.conf. Теперь давайте на базе этой конфигурации создадим диалплан. Диалпланы обоих серверов будут очень похожи, но для ясности здесь приведены оба. Новые строки, добавленные в существовавший до этого файл, выделены курсивом:

Файл extensions.conf для Торонто:

```
[globals]

[general]
autofallthrough=yes

[default]

[incoming_calls]
```

```
[phones]
include => internal
include => remote

[internal]
exten => _2XXX,1,NoOp()
exten => _2XXX,n,Dial(SIP/${EXTEN},30)
exten => _2XXX,n,Playback(the-party-you-are-calling&is-curntly-unavail)
exten => _2XXX,n,Hangup()

[remote]
exten => _1XXX,1,NoOp()
exten => _1XXX,n,Dial(SIP/osaka/${EXTEN})
exten => _1XXX,n,Hangup()

[osaka_incoming]
include => internal
```

Файл extensions.conf для Осаки:

```
[globals]

[general]
autofallthrough=yes

[default]

[incoming_calls]

[phones]
include => internal
include => remote

[internal]
exten => _1XXX,1,NoOp()
exten => _1XXX,n,Dial(SIP/${EXTEN},30)
exten => _1XXX,n,Playback(the-party-you-are-calling&is-curntly-unavail)
exten => _1XXX,n,Hangup()

[remote]
exten => _2XXX,1,NoOp()
exten => _2XXX,n,Dial(SIP/toronto/${EXTEN})
exten => _2XXX,n,Hangup()

[toronto_incoming]
include => internal
```

После того как файл extensions.conf сконфигурирован, можно выполнить его перезагрузку из консоли Asterisk с помощью команды `dialplan reload`. Удостовериться в том, что диалплан загружен, поможет команда `dialplan show`.

Вот и все! Теперь можно звонить с одного сервера Asterisk на другой.

# Конфигурация программного телефона IAX

Протокол IAX2 создан для обеспечения удобства работы с сетями, имеющими необычную конфигурацию, особенно с использованием технологии NAT (Network Address Translation – трансляция сетевых адресов). Это является его основным преимуществом и делает IAX2 превосходным протоколом для программных телефонов, выступающих в роли клиентов, поскольку они часто используются на портативных компьютерах, которые подключаются к различным сетям, зачастую без возможности управления самой сетью (например, при подключении к сети в разных гостиницах).

Протокол Inter-Asterisk eXchange (IAX) обычно используется для связи сервер–сервер; по сравнению с SIP его поддерживает большее число аппаратных телефонов. Есть и программные телефоны, поддерживающие протокол IAX, и работа по обеспечению поддержки аппаратных телефонов во встроенном ПО продолжается по нескольким направлениям. Основное различие между протоколами IAX и SIP – способ передачи медиа-данных между конечными точками.

При использовании протокола SIP для передачи трафика RTP (голоса) используются порты, отличные от тех, что работают с методами обмена сигналами. Например, Asterisk получает сигналы SIP через порт 5060, а трафик RTP (голос) проходит через порты от 10000 до 20000 по умолчанию. IAX-протокол отличается тем, что и обмен сигналами, и трафик медиа-данных выполняется через один порт: 4569. Следствие такого подхода – протокол IAX лучше подходит для топологий с использованием NAT.

Существует множество программных телефонов на базе IAX, но не так много аппаратных. Наиболее очевидная причина этому – IAX2 до сих пор не стандартизован IETF (Internet Engineering Task Force – Комитет по стандартизации интернет-протоколов), хотя многие уже перешли на него и пользуются предоставляемыми им преимуществами.

Превосходный программный телефон на базе IAX2 – idefisk. Он доступен бесплатно для скачивания по адресу *http://www.asteriskguru.com*[1]. Авторы данной книги добились замечательных результатов с этим программным телефоном, и, поскольку он выполняется в Microsoft Windows, Mac OS X и Linux, он является отличным примером для применения межплатформенных программных телефонов. Здесь будет продемонстрирована версия 1.31, хотя в апреле 2007 была выпущена версия 2.0, но пока не вышла ее реализация для Linux.

## Настройка конфигурационного файла канала (iax.conf)

Как всегда, постараемся наладить все быстро с минимальной настройкой конфигурационного файла, чтобы максимально сократить про-

[1] На сайте Asterisk Guru можно также найти большое количество превосходной документации!

блемы, которые могут возникнуть при конфигурации устройств. Как это было с файлом sip.conf, для регистрации IAX-телефона в Asterisk, в файл iax.conf необходимо добавить лишь несколько простых строк. Давайте посмотрим:

```
[general]
autokill=yes

[idefisk]
type=friend
host=dynamic
context=phones
```

Да, действительно, это все, что необходимо для настройки программного телефона. Это не самая безопасная или функциональная конфигурация (даже не используется пароль), но она будет работать.

В разделе `[general]` файла iax.conf имеется единственная опция – `autokill=yes`. Она используется для того, чтобы предотвратить задержку в системе, когда участник сети не отвечает (`ACK`) на пакет `NEW` (запрос на установление нового соединения) в течение 2000 мс. Вместо значения `yes` здесь можно задать время (в миллисекундах) ожидания `ACK` на пакет `NEW`. Управлять опцией `autokill` (автоуничтожение) для каждого отдельного равноправного участника сети можно, определяя параметр `qualify` (качество) для тех участников, о возможном недостаточном качестве используемых сетевых соединений которых известно заранее.

Весь остальной файл – описание программного телефона. Мы определяем для него тип `friend`, указывая Asterisk на то, что будем производить звонки на это устройство, а также принимать звонки с него. `friend` является сокращенной записью для одновременного определения `peer` (отправляющего вызовы программному телефону) и `user` (принимающего вызовы от программного телефона). Можно было бы дать отдельные описания для `peer` и `user`:

```
[idefisk]
type=user
context=phones

[idefisk]
type=peer
host=dynamic
```

Сконфигурировав файл iax.conf, сохраняем его и перезагружаем модуль канала IAX2 из консоли Asterisk с помощью команды `module reload chan_iax2.so`. Подтверждаем существование нового равноправного участника сети, выполнив команду `iax2 show peers`.

```
localhost*CLI> iax2 show peers
Name/Username  Host              Mask               Port Status
idefisk        (Unspecified) (D) 255.255.255.255    0 Unmonitored
1 iax2 peers [0 online, 0 offline, 1 unmonitored]
```

## Конфигурация программного телефона

После установки программного телефона idefisk откройте клиентское приложение. На экран будет выведено окно, представленное на рис. 4.6.


***Рис. 4.6.*** *Программный телефон idefisk*

После запуска программного телефона, чтобы можно было выполнять с него звонки, его необходимо настроить. Также, чтобы можно было принимать звонки, этот телефон должен быть зарегистрирован в Asterisk. Для этого щелкните правой кнопкой мыши по иконке в верхнем левом углу экрана. Откроется меню. Выберите в нем пункт Account Options (Опции учетной записи), что обеспечит открытие окна, показанного на рис. 4.7.

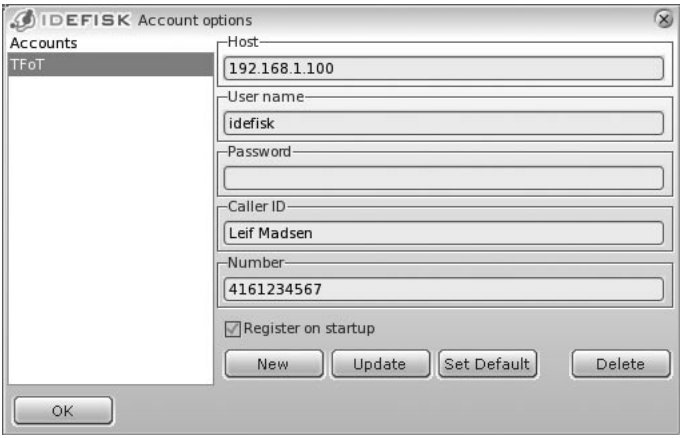



***Рис. 4.7.*** *Окно Account Options (Опции учетной записи) программного телефона idefisk*

Начните с создания новой учетной записи для программного телефона, щелкнув по кнопке New (Новая) и введя соответствующую информацию. В поле Host (Хост) должен быть указан IP-адрес или доменное имя вашей системы Asterisk, при этом имя пользователя должно совпадать с именем, указанным в квадратных скобках [] в файле iax.conf. Поле Password (Пароль) оставляем незаполненным, потому что мы не задавали параметр secret в файле iax.conf, а в полях Caller ID (ID звонящего) и Number (Номер) можно задать любые значения. Чтобы idefisk зарегистрировал эту учетную запись при запуске, поставьте флажок Register on startup (Зарегистрировать при запуске). Введя все необходимые данные, щелкните по кнопке OK, чтобы сохранить новую учетную запись.

Если был установлен флажок Register on startup (Зарегистрировать при запуске), телефон попытается зарегистрироваться в Asterisk. В консоли Asterisk будет выведена информация о том, что телефон зарегистрирован:

```
-- Registered IAX2 'idefisk' (UNAUTHENTICATED) at 127.0.0.1:32771
```

Проверить регистрацию можно из консоли Asterisk с помощью команды iax2 show peers:

```
localhost*CLI> iax2 show peers
Name/Username  Host          Mask            Port     Status
idefisk        127.0.0.1 (D) 255.255.255.255 32771    Unmonitored
1 iax2 peers [0 online, 0 offline, 1 unmonitored]
```

## Конфигурация диалплана для тестирования

Осталось только подтвердить возможность выполнения вызовов с помощью нашего телефона, задав простой диалплан в файле extensions.conf. Можно просто проверить наличие аудиосигнала в обоих направлениях, позвонив на добавочный номер 500, или настроить диалплан, созданный в разделе «Настройка диалплана для выполнения тестовых вызовов» данной главы, чтобы выполнить ряд тестовых вызовов. Если добавочный номер 1000 задан для SIP-телефона, что мы делали в предыдущих разделах, обеспечим, чтобы данная конфигурация не перекрывала его, и используем добавочный номер 1001 (если вы сконфигурировали несколько добавочных номеров для SIP, просто задайте здесь для программного телефона IAX2 уникальный добавочный номер):

```
[globals]

[general]

[default]
exten => s,1,Verbose(1|Unrouted call handler)
exten => s,n,Answer()
exten => s,n,Wait(1)
exten => s,n,Playback(tt-weasels)
exten => s,n,Hangup()

[incoming_calls]
```

```
[internal]
exten => 500,1,Verbose(1|Echo test application)
exten => 500,n,Echo()
exten => 500,n,Hangup()
exten => 1001,1,Verbose(1|Extension 1000)
exten => 1001,n,Dial(IAX2/idefisk,30)
exten => 1001,n,Hangup()

[phones]
include => phones
```

## Подключение к поставщику сервисов IAX

Некоторые поставщики сервисов интернет-телефонии (ITSP) предоставляют возможность начинать и завершать соединения с помощью протокола IAX2. Кроме сведения до минимума количества портов, которые необходимо открыть в межсетевом экране (для IAX2 требуется лишь один порт, через который ведется и обмен сигналами, и передача медиа-данных), способность объединения каналов этого протокола привлекательна как для поставщиков сервисов, так и для их клиентов из-за сохранения полосы пропускания, которое возможно при выполнении множества одновременных соединений между конечными точками.

Если ITSP предлагает завершение соединения с использованием IAX2, очень высока вероятность, что он использует Asterisk; таким образом, конфигурация для подключения к этому поставщику сервисов, скорее всего, будет аналогична той, которую мы приводим здесь.

Следующая конфигурация – это шаблон для подключения к поставщику сервисов IAX2:

```
[general]
autokill=yes

register => имяпользователя:пароль@мой.провайдер-сервиса.tld

[мой_уникальный_id]
type=user
secret=мой_уникальный_пароль
context=incoming_calls
trunking=yes
disallow=all
allow=gsm
allow=ulaw
deny=0.0.0.0/0.0.0.0
permit=10.251.100.1/255.255.255.255

[мой_уникальный_id]
type=peer
host=10.251.100.1
```

```
trunking=yes
disallow=all
allow=gsm
allow=ulaw
```

Чтобы принимать входящие вызовы по прямому номеру (номеру прямого набора внутренних абонентов – Direct Inward Dialing, DID), присвоенному вам поставщиком сервисов, необходимо откорректировать файл extensions.conf. Возможно, вы хотите направлять звонки на автоответчик или просто на телефон у себя на столе. В любом случае звонки от поставщика услуг можно принимать и сопоставлять с входящим DID с помощью следующей реализованной в диалплане логики:

```
[globals]

[general]
autofallthrough=yes

[default]

[incoming_calls]
exten => 14165551212,1,NoOp()
exten => 14165551212,n,Dial(SIP/1000,30)
exten => 14165551212,n,Playback(the-party-you-are-calling&is-curntly-unavail)
exten => 14165551212,n,Hangup()

exten => 4165551212,1,Goto(1${EXTEN})

[internal]

[phones]
include => internal
```

# Соединение двух серверов Asterisk по протоколу IAX

Часто желательно объединить два физических сервера Asterisk по протоколу IAX, чтобы иметь возможность обмениваться вызовами между двумя физическими местоположениями (расстояние между этими точками можем быть ничтожно мало, а может измеряться и километрами). Одно из преимуществ использования протокола IAX для этого – его способность, называемая *объединением каналов*, в которой используется метод отправки голосовых данных множества звонков под одним заголовком. Для одного или двух одновременных вызовов эффект от этой возможности невелик, но если между двумя точками выполняются десятки или сотни звонков, выигрыш в пропускной способности за счет использования объединения каналов может быть огромным.

### Конфигурация локального межсетевого экрана

Если ip-таблицы используются на одном компьютере с сервером Asterisk, можно выполнить следующие команды для открытия порта 4569 для протокола IAX2. Замечательная книга по организации межсетевых экранов с помощью ip-таблиц – «Linux Firewalls» (Novell Press) Стива Суэринга (Steve Suehring) и Роберта Циглера (Robert Ziegler).

```
# iptables -I RH-Firewall-1-INPUT -p udp --dport 4569 -j ACCEPT
# service iptables save
```

Помните, что это откроет порт 4569 для всего UDP-трафика из любого источника.

Системе понадобится интерфейс синхронизации – или аппаратный, производства Digium, или программный, использующий ядро драйвера ztdummy. Для этого в системе должен быть установлен и запущен драйвер Zaptel. Подробно об установке Zaptel рассказывается в главе 3.

## Конфигурация серверов Asterisk

Мы будем использовать простую схему из двух серверов Asterisk, зарегистрированных непосредственно друг на друге, и отдельных телефонов, зарегистрированных на каждом из серверов Asterisk. Будем называть серверы Asterisk Торонто и Осака (см. раздел «Соединение двух серверов Asterisk по протоколу SIP»). Телефон Боба будет зарегистрирован и подключен к Торонто, а телефон Элис – к серверу Осака.

Прежде всего создадим новый файл канала (iax.conf). Для этого переименуем текущий файл шаблона в iax.conf.sample и создадим новый пустой файл iax.conf:

```
# cd /etc/asterisk
# mv iax.conf iax.conf.sample
# touch iax.conf
```

Далее откроем файл iax.conf и введем следующие настройки для сервера Asterisk Торонто:

```
[general]
autokill=yes

register => toronto:welcome@192.168.1.107

[osaka]
type=friend
host=dynamic
trunk=yes
secret=welcome
```

```
context=incoming_osaka
deny=0.0.0.0/0.0.0.0
permit=192.168.1.107/255.255.255.255
```

Пояснения к параметру `autokill=yes` приведены в предыдущем разделе, а его назначение – гарантировать, что новые соединения, устанавливаемые с удаленной системой и не получившие подтверждения приема в течение заданного времени (по умолчанию – две секунды), корректно завершаются. Это спасает от возникновения множества подвешенных каналов, просто ожидающих подтверждения приема, которое, возможно, никогда не будет получено.

Строка `register` используется для указания удаленному серверу Asterisk нашего местоположения, чтобы, когда сервер по адресу 192.168.1.107 будет готов послать нам вызов, он отправлял его на наш IP-адрес (в данном случае наш IP-адрес – 192.168.1.104, его мы увидим в конфигурационном файле iax.conf сервера Осака). Имя пользователя `Toronto` и пароль `welcome` посылаются на сервер Осака, который проверяет нашу регистрацию. Если аутентификация пройдена успешно, он записывает в память местоположение нашего сервера Asterisk и будет использовать эту информацию при отправке нам вызовов.

Описание `[Osaka]` используется для управления аутентификацией удаленного сервера и доставки на него нашего диалплана. Osaka – это имя пользователя, используемое для аутентификации при поступлении вызовов. Для параметра `type` задано значение `friend`, потому что мы хотим иметь возможность отправлять вызовы на сервер Осака и принимать от него вызовы. Для параметра `host` задано значение `dynamic`, что указывает Asterisk направлять вызовы на IP-адрес, полученный при регистрации противоположной конечной точки.

В начале данного раздела был упомянут потенциальный выигрыш в пропускной способности при использовании возможности объединения каналов, предоставляемой IAX2. В Asterisk эту функциональность активировать просто, надо лишь добавить строку `trunk=yes` в описание сервера типа `friend`. Если интерфейс синхронизации (то есть dummy) установлен и запущен, можно использовать преимущества объединения каналов IAX2.

Параметр `secret` ясен – это пароль, используемый для аутентификации. Контекст `[incoming_osaka]` – это раздел файла extensions.conf, где будут обрабатываться входящие звонки для этого сервера (`friend`). Наконец, параметр `deny` запрещает все IP-адреса, кроме явно разрешенного 192.168.1.107.

Конфигурация iax.conf для Осаки практически идентична, за исключением IP-адреса и имен:

```
[general]
autokill=yes

register => osaka:welcome@192.168.1.104
```

```
[toronto]
type=friend
host=dynamic
trunk=yes
secret=welcome
context=incoming_toronto
deny=0.0.0.0/0.0.0.0
permit=192.168.1.104/255.255.255.255
```

## Конфигурация телефона IAX

В разделе «Конфигурация программного телефона» мы занимались настройкой нашего первого программного телефона IAX2 на примере idefisk. Конфигурация, которая будет использоваться здесь, практически аналогична, за исключением незначительных изменений, связанных с необходимостью обеспечения уникальности участников сети. Если вы уже настроили программный телефон SIP, можете также использовать его в качестве одного (или обоих) равноправных участников сети. Не забывайте, что Asterisk – приложение, работающее с множеством протоколов, и вызов к Asterisk может быть послан с SIP-телефона, передан по магистральному каналу IAX2 и затем направлен на другой SIP-телефон (или H.323, MGCP и т. д.).

Для Осаки:

```
[1001]
type=friend
host=dynamic
context=phones
```

Для Торонто:

```
[2001]
type=friend
host=dynamic
context=phones
```

Далее сконфигурируем программный телефон IAX2 для регистрации в Asterisk. Если телефон успешно зарегистрирован, в командной строке будет выведено примерно следующее сообщение:

```
*CLI>    -- Registered IAX2 '1001' (UNAUTHENTICATED) at 192.168.1.104:4569
```

## Конфигурация диалплана

Чтобы обеспечить возможность установления соединения между двумя серверами Asterisk по магистральному каналу IAX2, необходимо сконфигурировать простой диалплан. Представленный диалплан будет направлять вызовы на все добавочные номера в диапазоне 1000 (от 1000 до 1999) на сервер Осака и вызовы на все добавочные номера в диапазоне 2000 (от 2000 до 2999) на сервер Торонто. В данном примере предполагается, что сконфигурирована пара программных телефонов IAX2, но если в вашем распоряжении имеется SIP-телефон (или

два), можно использовать и его. Просто помните, что в этом случае понадобится изменить приложение `Dial()` так, чтобы вызовы направлялись на SIP-телефон по SIP-протоколу, а не IAX2 (то есть строку `Dial(IAX2/${EXTEN},30)` необходимо заменить на `Dial(SIP/${EXTEN},30)`).

Файл extensions.conf для Торонто:

```
[globals]

[general]
autofallthrough=yes

[default]

[incoming_calls]

[phones]
include => internal
include => remote

[internal]
exten => _1XXX,1,NoOp()
exten => _1XXX,n,Dial(IAX2/${EXTEN},30)
exten => _1XXX,n,Playback(the-party-you-are-calling&is-curntly-unavail)
exten => _1XXX,n,Hangup()

[remote]
exten => _2XXX,1,NoOp()
exten => _2XXX,n,Dial(IAX2/toronto/${EXTEN})
exten => _2XXX,n,Hangup()

[toronto_incoming]
include => internal
```

Файл extensions.conf для Осаки:

```
[globals]

[general]
autofallthrough=yes

[default]

[incoming_calls]

[phones]
include => internal
include => remote

[internal]
exten => _2XXX,1,NoOp()
exten => _2XXX,n,Dial(IAX2/${EXTEN},30)
exten => _2XXX,n,Playback(the-party-you-are-calling&is-curntly-unavail)
exten => _2XXX,n,Hangup()
```

```
[remote]
exten => _1XXX,1,NoOp()
exten => _1XXX,n,Dial(IAX2/osaka/${EXTEN})
exten => _1XXX,n,Hangup()

[osaka_incoming]
include => internal
```

## Использование шаблонов в конфигурационных файлах

С конфигурационными файлами Asterisk связан один очень малоизвестный факт, но он настолько замечательный, что заслуживает отдельного небольшого раздела.

Скажем, имеется 20 SIP-телефонов, практически идентичных с точки зрения конфигурации. Согласно документации они должны описываться путем задания параметров для каждого телефона в отдельности. Фрагмент подобного файла sip.conf мог бы выглядеть так:

```
[1000]
type=friend
context=internal
host=dynamic
disallow=all
allow=ulaw
dtmfmode=rfc2833
maibox=1000
secret=AllYourSetsAreBelongToUs

[1001]
type=friend
context=internal
host=dynamic
disallow=all
allow=ulaw
dtmfmode=rfc2833
maibox=1001
secret=AllYourSetsAreBelongToUs

[1002]
type=friend
context=internal
host=dynamic
disallow=all
allow=ulaw
dtmfmode=rfc2833
maibox=1002
secret=AllYourSetsAreBelongToUs
```

Слишком много ввода текста, копирования и вставки, правда? А что если требуется изменить имя контекста для телефонов. Не очень удобно, не так ли?

Вводим шаблон. Давайте создадим таких же участников сети типа `friend`, как делали выше, только на этот раз используя шаблон:

```
[sets](!) ; <== обратите внимание, восклицательный знак
          ; взят в круглые скобки. Это признак шаблона.
type=friend
context=internal
host=dynamic
disallow=all
allow=ulaw
dtmfmode=rfc2833
secret=AllYourSetsAreBelongToUs

[1000](sets) ; <== обратите внимание, имя шаблона взято
             ; в круглые скобки. Все настройки этого шаблона
             ; будут унаследованы.
maibox=1000

[1001](sets)
maibox=1001

[1002](sets)
maibox=1002
```

Это одна из самых малоизвестных возможностей создания конфигурационного файла. Очень немногие пользуются этой возможностью, но лишь потому, что мало кто знает о ней. Итак, пришло время перемен. С этого момента мы хотим видеть, что шаблонами пользуются все; и да, мы будем проверять.

# Отладка

В Asterisk предлагается несколько методов отладки. Подключившись к консоли, можно управлять отладкой и уровнем детальности сообщений, а также трассировкой пакетов протокола. В данном разделе рассмотрим доступные варианты.

## Подключение к консоли

Чтобы подключиться к консоли Asterisk, можно или запустить сервер непосредственно из консоли (в этом случае невозможно будет выйти из консоли, не завершив работу Asterisk), или запустить Asterisk как демон и затем подключиться к удаленной консоли.

Чтобы запустить процесс Asterisk непосредственно из консоли, используйте флаг консоли:

```
# /usr/sbin/asterisk -c
```

Чтобы подключиться к удаленной консоли, сначала запустите демон, а затем выполните подключение, используя флаг `-r`:

```
# /usr/sbin/asterisk
# /usr/sbin/asterisk -r
```

Если какой-то модуль не загружается или Asterisk не загружается из-за какого-то модуля, запустите Asterisk с флагом `-c`, чтобы отслеживать статус загружаемых модулей. Например, если при попытке загрузить драйвер канала OSS (который позволяет использовать канал `CONSOLE` (консоль)) Asterisk не может открыть /dev/dsp, при запуске будет получено сообщение о такой ошибке:

```
WARNING[32174]: chan_oss.c:470 soundcard_init: Unable to open /dev/dsp:
No such file or directory
== No sound card detected -- console channel will be unavailable
== Turn off OSS support by adding 'noload=chan_oss.so' in /etc/asterisk/
   modules.conf
```

WARNING[32174]: chan_oss.c:470 soundcard_init: Не получается открыть /dev/dsp:

Файл или каталог не существует

== Звуковая карта не найдена – канал консоли будет недоступен

== Отключите поддержку OSS, добавив 'noload=chan_oss.so' в /etc/asterisk/modules.conf

## Изменение детальности сообщений и включение отладки

Asterisk может выводить отладочную информацию в форме сообщений `WARNING` (предупреждение), `NOTICE` (извещение) и `ERROR` (ошибка). Эти сообщения предоставляют информацию о системе, такую как регистрационные данные, статус, последовательность вызовов и другие полезные сведения. Обратите внимание, что сообщения `WARNING` и `NOTICE` не являются сообщениями об ошибках; а вот к сообщениям `ERROR` необходимо относиться внимательно. Уровень детальности сообщений можно задать с помощью команды `set verbose` и числового значения. Диапазон допустимых значений – от 3 до 10. Например, чтобы задать самый высокий уровень детальности, используется команда

```
# set verbose 10
```

Также можно активировать вывод основных сообщений отладки с помощью команды `set debug` и числового значения. Чтобы активировать вывод сообщений `DEBUG` (отладка) в консоли, возможно, понадобится в файле logger.conf добавить `debug` в выражение `console =>`:

```
console => warning,notice,error,event,debug
```

Диапазон допустимых значений для `set debug` – от 3 до 10. Например:

```
# set debug 10
```

# Заключение

Проработав все разделы данной главы, вы должны получить пару сконфигурированных аналоговых интерфейсов, локальные каналы SIP и IAX2, подключенные к программному и/или аппаратному телефону, и иметь возможность передавать вызовы через серверы по протоколам SIP и IAX2. Эти настройки элементарны, но благодаря им мы имеем функциональные каналы, с которыми можно работать. Мы будем использовать их в следующих главах, пока не научимся создавать более функциональные диалпланы.

# 5

# Основы диаллана

*Все должно быть сделано настолько просто,*
*насколько это возможно, но не проще.*

– Альберт Эйнштейн (1879–1955)

Диалплан, поистине, – сердце любой системы Asterisk, поскольку он определяет, как Asterisk обрабатывает входящие и исходящие вызовы. По сути, он состоит из списка инструкций или шагов, которым будет следовать Asterisk. В отличие от традиционных систем телефонной связи, диалплан Asterisk является полностью настраиваемым. Чтобы добиться успеха в построении собственной системы Asterisk, необходимо понять концепцию диалплана.

Если вы пытались прочитать некоторые примеры диалпланов и сочли их невыполнимыми или пробовали написать диалплан Asterisk и не достигли успеха, не отчаивайтесь, помощь близка. Данная глава шаг за шагом объяснит, как работает диалплан. Прочитав ее, вы приобретете навыки, необходимые для создания собственного диалплана. Все примеры дополняют друг друга, поэтому можно свободно вернуться назад и перечитать раздел, если что-то непонятно. Однако, пожалуйста, обратите внимание, что данная глава ни в коем случае не является исчерпывающим обзором всех возможностей диалплана; наша цель – рассмотреть только основы. Более глубокие аспекты настройки диалплана будут затронуты в последующих главах.

## Синтаксис диалплана

Диалплан Asterisk определен в конфигурационном файле extensions.conf.

Файл extensions.conf обычно находится в папке /etc/asterisk/, но его местоположение может меняться в зависимости от того, как установлена Asterisk. Этот файл часто размещается в папках /usr/local/asterisk/etc/ и /opt/asterisk/etc/.

Диалплан состоит из четырех основных элементов: контекстов, добавочных номеров, приоритетов и приложений. В следующих нескольких разделах будут рассмотрены все эти части и то, как они работают вместе. После объяснения роли каждого из этих элементов в диалплане перейдем к процессу поэтапного создания базового функционального диалплана.

### Образцы конфигурационных файлов

Если при установке Asterisk были установлены и образцы конфигурационных файлов, скорее всего, у вас имеется файл extensions.conf. Но мы предлагаем начинать не с файла-образца, а создать собственный файл extensions.conf с нуля. Это будет крайне полезно, поскольку обеспечит лучшее понимание основных элементов и принципов диалплана.

Но надо сказать, что образец файла extensions.conf остается фантастическим ресурсом, полным примеров и идей, которые можно использовать после изучения базовых элементов. Мы предлагаем переименовать его и назвать, например, extensions.conf.sample. Таким образом оригинальный файл будет сохранен, и к нему можно будет обратиться в будущем. Образцы конфигурационных файлов также можно найти в подпапке /configs/ папки исходного кода Asterisk.

## Контексты

Диалпланы разбиты на разделы, называемые *контекстами*. Контексты – это именованные группы добавочных номеров, которые выполняют несколько функций.

Контексты изолируют разные части диалплана, предотвращая возможность их взаимодействия. Добавочный номер, определенный в одном контексте, полностью изолирован от добавочных номеров другого контекста, если только взаимодействие не разрешено специально. (То, как разрешить взаимодействие контекстов, будет рассмотрено ближе к концу данной главы.)

В качестве простого примера возьмем две компании, совместно использующие один сервер Asterisk. Если разместить голосовое меню каждой компании в собственном контексте, они будут эффективно отделены друг от друга. Это позволяет независимо определять действия, выполняемые, скажем, при наборе номера 0: при нажатии кнопки 0 в голосовом меню компании A вы попадете к секретарю компании A, а при на-

жатии кнопки 0 в голосовом меню компании B – к секретарю компании B. (Эти примеры предполагают, конечно же, что мы задали для Asterisk перенаправление вызовов на секретарей при нажатии кнопки 0 вызывающими абонентами.)

Контексты различаются по именам. Имена контекстов заключаются в квадратные скобки ([ ]). Допустимыми символами для образования имени являются буквы от A до Z (верхнего и нижнего регистра), цифры от 0 до 9, дефис и символ подчеркивания[1]. Например, контекст для входящих вызовов выглядит так:

```
[incoming]
```

Максимальная длина имени контекста – 79 символов (80 символов – 1 завершающий нуль).

Все инструкции, размещаемые после описания контекста и до описания следующего контекста, являются частью данного контекста. В начале диалплана находятся два специальных контекста, `[general]` и `[globals]`. Раздел `[general]` содержит список общих настроек диалплана (о которых, вероятно, вам никогда не придется беспокоиться), а контекст `[globals]` мы будем обсуждать в разделе «Глобальные переменные». Пока что достаточно знать, что эти два контекста являются специальными. Созданные вами контексты можно называть как угодно, только не используйте имена `[general]` и `[globals]`.

При описании канала (а именно так выполняется подключение элементов к системе) одним из параметров этого описания является контекст. Иначе говоря, контекст – это точка диалплана, с которой будет начинаться обработка соединений, выполняемых через этот канал.

Другое важное применение контекстов (возможно, самое важное) – обеспечение безопасности. Правильно применяя контексты, определенным абонентам можно предоставить доступ к функциям (таким, как междугородная связь), которые недоступны для других. Если диалплан разработан неаккуратно, пользователи по вашей оплошности могут получить возможность мошенничать в вашей системе. Пожалуйста, помните об этом при построении системы Asterisk.

В подпапке doc/ папки исходного кода Asterisk находится очень важный файл, security.txt, в котором определено несколько шагов для обеспечения безопасности системы Asterisk. Вам чрезвычайно важно прочитать и понять этот файл. Если проигнорировать меры предосторожности, описанные там, все может закончиться тем, что кто угодно сможет делать междугородные и платные звонки за ваш счет!

[1] Пожалуйста, обратите внимание, что пробел не входит в список допустимых символов. Не используйте пробелы в именах контекстов, потому что результат вам не понравится!

Если не отнестись к безопасности своей системы Asterisk серьезно, может дойти до того, что вам придется за это дорого расплатиться в буквальном смысле этого слова! *Пожалуйста*, выделите время и силы на то, чтобы защитить свою систему от мошенничества с оплатой.

## Добавочные номера

В мире телекоммуникаций термин «*добавочный номер*» (extension) обычно обозначает числовой идентификатор, который присвоен линии, идущей к конкретному телефону. Однако в Asterisk это намного более широкое понятие, поскольку оно определяет уникальные последовательности шагов (каждый шаг включает приложение), которые Asterisk будет применять к вызову по этой линии. В каждом контексте может быть задано столько добавочных номеров, сколько требуется. При вызове конкретного добавочного номера (входящим или внутренним звонком) Asterisk будет выполнять шаги, определенные для этого добавочного номера. Поэтому именно добавочные номера определяют, что происходит со звонками при их обработке соответственно диалплану. Хотя, конечно, добавочные номера могут использоваться в их традиционном значении (то есть вызов добавочного номера 153 заставит зазвонить SIP-телефон на столе Джона), но в диалплане Asterisk они могут означать намного большее.

Синтаксис добавочного номера – это слово `exten`, за которым следует стрелка, образованная знаками равенства и «больше чем»:

```
exten =>
```

Далее указывается имя (или номер). В традиционных системах телефонной связи под добавочными номерами мы понимаем цифры, которые надо набрать, чтобы другой телефон зазвонил. В Asterisk это понятие намного шире; например, в качестве имени добавочного номера может использоваться любая комбинация цифр и букв. В данной главе и далее будут использоваться как цифровые, так и буквенно-цифровые добавочные номера.

Присвоение имен добавочным номерам может показаться революционной идеей, но если вспомнить, что многие транспортные протоколы VoIP поддерживают (или даже активно поощряют) вызовы по имени или адресу электронной почты, а не просто по номеру, это действительно имеет смысл. Это одно из свойств, делающих Asterisk такой гибкой и мощной системой.

Полный добавочный номер состоит из трех компонентов:

- Имени (или номера).
- Приоритета (каждый добавочный номер может включать множество шагов; порядковый номер шага называется его приоритетом).

- Приложения (или команды), которое выполняет некоторое действие над вызовом.

Эти три компонента разделяются запятыми:

```
exten => имя,приоритет,приложение()
```

Вот пример того, как может выглядеть настоящий добавочный номер:

```
exten => 123,1,Answer()
```

В этом примере имя добавочного номера – `123`, приоритет – `1`, а приложение – `Answer()`. Теперь пойдем дальше и рассмотрим, что такое приоритеты и приложения.

## Приоритеты

Каждый добавочный номер может включать множество шагов, называемых *приоритетами*. Каждый приоритет пронумерован последовательно, начиная с 1, и выполняет одно определенное приложение. Например, следующий добавочный номер отвечает на звонок (в приоритете под номером 1) и затем выполняет разъединение (в приоритете под номером 2):

```
exten => 123,1,Answer()
exten => 123,2,Hangup()
```

Не переживайте, если вы не понимаете, что такое `Answer()` и `Hangup()`, мы очень скоро их рассмотрим. Здесь главное – запомнить, что для отдельного добавочного номера Asterisk выполняет приоритеты по порядку.

### Ненумерованные приоритеты

В более старых версиях Asterisk нумерация приоритетов вызывала множество проблем. Представьте, что в добавочном номере 15 приоритетов и требуется добавить что-то в шаге 2. Номера всех последующих приоритетов пришлось бы менять вручную. Asterisk не обрабатывает пропущенные шаги или неправильно пронумерованные приоритеты, и отладка ошибок такого типа превращалась в бесцельную и досадную трату времени.

Начиная с версии 1.2 Asterisk решила эту проблему. Был введен приоритет `n`, что означает «следующий». Каждый раз, когда Asterisk встречает приоритет `n`, она берет номер предыдущего приоритета и добавляет 1. Это упрощает внесение изменений в диалплан, поскольку теперь не надо изменять номера всех шагов. Например, диалплан может быть таким:

```
exten => 123,1,Answer()
exten => 123,n,выполнить что-то
exten => 123,n,выполнить что-то еще
exten => 123,n,выполнить последнее
exten => 123,n,Hangup()
```

Asterisk будет самостоятельно вычислять номер следующего приоритета при каждой встрече с приоритетом `n`[1]. Однако следует отметить, что приоритет под номером 1 должен быть задан *обязательно*. Если случайно для идущего первым приоритета задать `n` вместо `1`, добавочный номер будет недоступен.

### Метки приоритетов

Начиная с версии 1.2 в Asterisk стало общепринятой практикой присваивать приоритетам текстовые метки. Это обеспечивает возможность ссылаться на приоритет не по номеру, который может быть неизвестен, потому что теперь в диалпланах, как правило, используются ненумерованные приоритеты. Чтобы присвоить приоритету текстовую метку, просто добавляем ее в круглых скобках после приоритета:

```
exten => 123,n(метка),приложение()
```

Очень распространенной ошибкой является использование запятой между символами `n` и `(`, как в данном примере:

```
exten => 123,n,(метка),приложение() ;<-- ЭТО НЕ БУДЕТ РАБОТАТЬ
```

Это приведет к нарушению данной части диалплана, будет выдано сообщение об ошибке, из-за того что приложение не найдено.

В следующей главе мы рассмотрим, как переходить с одного приоритета на другой, используя логику диалплана. Вы будете встречать множество меток приоритетов и станете очень часто использовать их в своих диалпланах.

## Приложения

Приложения – это рабочие лошадки диалплана. Каждое приложение выполняет определенное действие над данным каналом, например воспроизведение звука, прием тонального ввода, вызов канала, разрыв соединения и т. д. В предыдущем примере было представлено два простых приложения: `Answer()` и `Hangup()`. Сейчас мы подробнее рассмотрим, как они работают.

Для выполнения некоторых приложений, таких как `Answer()` и `Hangup()`, не требуется никаких дополнительных инструкций. Некоторым приложениям необходима дополнительная информация. Эти данные, называемые *аргументами*, могут передаваться в приложения, чтобы

[1] Asterisk допускает выполнение простых арифметических операций с приоритетами, таких как `n + 200`, или использование приоритета `s` (от английского same – такой же), но для их применения вам нужно быть профессионалом. Будьте добры, обратите внимание, что добавочный номер `s` и приоритет `s` – это две разные вещи.

оказывать влияние на то, как они выполняют свои действия. Чтобы передать аргументы в приложение, разместите их через запятую в круглых скобках, следующих за именем приложения.

Иногда вместо запятой в качестве разделителя между аргументами можно увидеть символ вертикальной черты (|). Допускается использование любого из этих символов. В примерах данной книги для разделения аргументов приложения будет применяться запятая, поскольку авторы предпочитают такой синтаксис. Однако при синтаксическом разборе диалплана Asterisk преобразует все запятые в аргументах приложений в символы вертикальной черты.

Когда мы создадим наш первый диалплан в следующем разделе, вы научитесь использовать приложения и связанные с ними аргументы.

# Простой диалплан

Теперь мы готовы создать наш первый диалплан. Давайте начнем с очень простого примера. Asterisk должна будет ответить на звонок, воспроизвести звуковой файл и разорвать соединение. Используем этот простой пример, чтобы обозначить наиболее важные концепции диалплана.

Предложенные в данной главе примеры разработаны исходя из предположения, что был создан и сконфигурирован (соответственно описанию в предыдущей главе) по крайней мере один канал (Zap, SIP или IAX2 – неважно) и что все вызовы, поступающие на этот канал, направляются в контекст диалплана `[incoming]`. Если к какому-то из предыдущих примеров вы подошли творчески, вероятно, вам придется вносить некоторые поправки, чтобы обеспечить соответствие имен каналов.

## Добавочный номер s

В наших каналах мы применяем определенную технологию, и поэтому, прежде чем приступить к настройке диалплана, придется остановиться еще на одном вопросе. Необходимо рассмотреть добавочный номер `s`. Когда в контекст поступают вызовы, для которых не указан конкретный добавочный номер (например, вызов FXO-линии), они передаются на добавочный номер `s`. (`s` – сокращение от start (начало), поскольку именно здесь начнется обработка вызова, если с ним не передана информация о добавочном номере.)

Поскольку это именно то, что требуется для нашего диалплана, перейдем к делу. Для каждого вызова будет выполняться три действия (ответ на него, воспроизведение звукового файла и разъединение), поэтому добавочному номеру `s` понадобится три приоритета. Поместим

три приоритета в контекст [incoming], поскольку было принято решение о том, что все входящие вызовы должны обрабатываться в этом контексте[1].

```
[incoming]
exten => s,1,приложение()
exten => s,n,приложение()
exten => s,n,приложение()
```

Теперь осталось только вставить приложения – и наш первый диалплан готов.

Обратите внимание, что можно было бы пронумеровать каждый приоритет, как показано ниже, но теперь так делать не рекомендуется, поскольку это сильно усложняет внесение изменений в диалплан впоследствии:

```
[incoming]
exten => s,1,приложение()
exten => s,2,приложение()
exten => s,3,приложение()
```

## Приложения Answer(), Playback() и Hangup()

Прежде чем мы будем отвечать на звонок, воспроизводить звуковой файл и затем выполнять разъединение вызова, нам нужно научиться это делать. Приложение Answer() (Ответ) используется для ответа каналу, по которому выполняется звонок. Оно выполняет исходную настройку для канала, получающего входящий вызов. (Некоторые приложения не требуют обязательного ответа каналу, но соответствующий ответ на звонок перед тем, как выполнять какие-либо действия над каналом, является очень хорошей практикой.) Как упоминалось ранее, Answer() не принимает аргументов.

Приложение Playback() (Воспроизведение) воспроизводит в канале предварительно записанный звуковой файл. При использовании приложения Playback() ввод, поступающий от пользователя, просто игнорируется.

---

[1] В имени контекста нет ничего особенного. Данный контекст мог бы называться [stuff_that_comes_in] (все_что_поступает), и, поскольку контекст назначается в описании канала в файлах sip.conf, iax.conf, zaptel.conf и других, канал перешел бы в этот контекст диалплана. При этом настоятельно рекомендуется присваивать контекстам имена, из которых можно понять их назначение. Хорошими именами контекстов были бы [incoming], [local_calls], [long_distance], [sip_telephones], [user_services], [experimental], [remote_locations] и т. д. Нельзя забывать, что контекст определяет, как канал входит в диалплан, поэтому контекстам должны присваиваться соответствующие имена.

С Asterisk поставляется множество профессионально записанных звуковых файлов, которые должны находиться в папке, используемой для хранения звуков по умолчанию (обычно это /var/lib/asterisk/sounds/). При компиляции Asterisk можно выбрать для установки различные наборы образцов звуков, записанных на разных языках и в разных форматах файлов. Во многих примерах данной книги будут использоваться эти файлы, а также несколько файлов из Extra Sound Package, поэтому, пожалуйста, потратьте немного времени и установите этот пакет (см. главу 3). Также, посетив сайт *http://thevoice.digium.com/*, можно создать собственные голосовые сообщения, записанные тем же голосом, что и предоставляемые стандартные сообщения.

Чтобы использовать `Playback()`, задайте в качестве аргумента имя файла (без расширения). Например, `Playback(filename)` обеспечит воспроизведение звукового файла filename.gsm, предполагая, что он размещен в стандартной папке для звуковых файлов. Обратите внимание, что по желанию можно указать полный путь к файлу, как это сделано в данном примере:

```
Playback(/home/john/sounds/filename)
```

Этот пример обеспечит воспроизведение файла filename.gsm из папки /home/john/sounds/. Также можно использовать относительные пути из папки для звуковых файлов Asterisk:

```
Playback(custom/filename)
```

Этот пример обеспечит воспроизведение filename.gsm из подпапки custom/ стандартной папки для звуковых файлов (вероятно, это будет /var/lib/asterisk/sounds/custom/filename.gsm). Заметьте, что, если в указанной папке содержится несколько файлов под одним именем, но с разными расширениями файлов, Asterisk автоматически воспроизводит лучший из них[1].

Приложение `Hangup()` (Разъединить) выполняет именно то, что подразумевается под его именем: оно разъединяет активный канал. Это приложение должно применяться в конце контекста для завершения текущего вызова, что защитит от несанкционированного использования диалплана абонентами. Приложение `Hangup()` не принимает аргументов.

[1] Asterisk выбирает лучший файл исходя из затрат на преобразование, то есть тот файл, для которого преобразование в собственный аудиоформат обусловит наименьшую нагрузку на ЦП. При запуске Asterisk вычисляет затраты на преобразования между разными аудиоформатами (часто в разных системах они различны). Величины этих затрат можно увидеть, введя команду `show translation` в интерфейсе командной строки. Представленные числа соответствуют времени в миллисекундах, которое потребуется Asterisk для преобразования одной секунды звука. Подробнее разные аудиоформаты (известные как кодеки) будут рассмотрены в главе 8.

## Наш первый диалплан

Теперь, когда добавочный номер готов, сведем все вместе и создадим наш первый диалплан. Как принято во многих технических книгах (особенно в книгах по программированию), первый пример будет называться «Hello World!» (Здравствуй, мир!).

В первом приоритете нашего добавочного номера мы будем отвечать на звонок, во втором – воспроизводить звуковой файл hello-world.gsm, а в третьем будет выполнен разрыв соединения. Вот как выглядит диалплан:

```
[incoming]
exten => s,1,Answer()
exten => s,n,Playback(hello-world)
exten => s,n,Hangup()
```

Если у вас уже имеется один или несколько сконфигурированных каналов – вперед[1]! Просто создайте файл extensions.conf (например, в папке /etc/asterisk) и вставьте в него четыре строки кода диалплана, которые мы только что написали. Если ничего не получается, проверьте, нет ли в консоли Asterisk сообщений об ошибках, и убедитесь, что для ваших каналов задан контекст `[incoming]`.

Даже несмотря на то что этот пример очень мал и прост, он раскрывает основные принципы контекстов, добавочных номеров, приоритетов и приложений. Если все получилось и этот диалплан заработал, значит, вы разобрались с основами, на базе которых создаются все диалпланы.

Теперь давайте дополнять наш пример. В конце концов, в телефонной системе, просто воспроизводящей звуковой файл и затем разъединяющей канал, очень немного пользы!

# Создание интерактивного диалплана

Созданный в предыдущем разделе диалплан был статическим; он всегда выполняет одни и те же действия для всех вызовов. Теперь мы собираемся добавить некоторую логику в диалплан, чтобы он осуществлял разные действия на основании ввода пользователя. Для этого необходимо рассмотреть еще некоторые приложения.

---

[1] Кстати, если у вас еще нет сконфигурированных каналов, самое время заняться ими. Вы испытаете настоящее удовольствие, когда впервые сделаете звонок в систему Asterisk, которую построили с нуля. Когда люди понимают, что только что они создали телефонную систему, на их лицах появляется такая забавная улыбка. Это удовольствие может быть доступным и вам, поэтому, пожалуйста, не двигайтесь дальше, пока не испытаете этот маленький диалплан в действии.

## Приложения Background(), WaitExten() и Goto()

Один из самых важных ключей к построению интерактивных диалпланов Asterisk – приложение `Background()`[1] (Фон). Как и `Playback()`, это приложение воспроизводит записанный звуковой файл. Однако, в отличие от `Playback()`, если пользователь нажимает кнопку (или последовательность кнопок) на клавиатуре телефона, оно прерывает воспроизведение и переходит к добавочному номеру соответственно нажатым цифрам. Например, если абонент нажмет кнопку 5, Asterisk прекратит воспроизводить звуковое сообщение и передаст управление вызовом первому приоритету добавочного номера 5.

Чаще всего приложение `Background()` используется для создания голосовых меню (которые часто называют *автоответчиками* или *интерактивными секретарями*). Многие компании используют голосовые меню для направления абонентов на соответствующий добавочный номер, таким образом освобождая своих секретарей от необходимости отвечать на все звонки.

Синтаксис `Background()` аналогичен синтаксису `Playback()`:

```
exten => 123,1,Answer()
exten => 123,n,Background(main-menu)
```

В более ранних версиях Asterisk, если приложение `Background()` завершало воспроизведение звукового сообщения и в текущем добавочном номере больше не было приоритетов, Asterisk ничего не делала и ожидала ввода абонента. Такое поведение больше не является для Asterisk принятым по умолчанию. Если требуется, чтобы Asterisk ожидала ввода абонента после завершения воспроизведения звукового сообщения, можно вызвать приложение `WaitExten()` (Ожидание добавочного номера). Приложение `WaitExten()` ожидает от абонента набора телефонного номера и часто вызывается сразу после приложения `Background()`, как в данном фрагменте диалплана:

```
exten => 123,1,Answer()
exten => 123,n,Background(main-menu)
exten => 123,n,WaitExten()
```

Если требуется, чтобы приложение `WaitExten()` ожидало ответа в течение определенного времени (вместо использования времени ожидания по умолчанию), просто укажите число, соответствующее необходимому количеству секунд, в качестве первого аргумента в `WaitExten()`:

```
exten => 123,n,WaitExten(5)
```

И `Background()`, и `WaitExten()` позволяют абоненту производить набор номера. После этого Asterisk пытается найти в текущем контексте доба-

[1] Нужно заметить следующее: некоторые люди ожидают, исходя из имени этого приложения, что `Background()` будет выполняться в диалплане до тех пор, пока воспроизводится звук, но это имя указывает на то, что данное приложение воспроизводит звук в фоновом режиме в процессе ожидания двухтонального многочастотного набора телефонного номера (Dual-Tone Multi-Frequency, DTMF).

вочный номер, соответствующий введенным абонентом цифрам. Если Asterisk находит однозначное соответствие, она направляет вызов на этот добавочный номер. Продемонстрируем это, добавив несколько строк в наш пример:

```
exten => 123,1,Answer()
exten => 123,n,Background(main-menu)
exten => 123,n,WaitExten()

exten => 2,1,Playback(digits/2)

exten => 3,1,Playback(digits/3)

exten => 4,1,Playback(digits/4)
```

Если вызвать добавочный номер 123 из примера выше, он воспроизведет звуковое сообщение с фразой «main menu» (главное меню) и после этого будет ожидать ввода цифр 2, 3 или 4. Если нажать одну из этих цифр, Asterisk воспроизведет ее для вас. Также вы обнаружите, что, если ввести другую цифру (например, 5), Asterisk не обеспечит ожидаемого результата.

Также возможна ситуация, когда Asterisk обнаружит неоднозначное соответствие. Это можно легко продемонстрировать, введя в предыдущий пример добавочный номер под именем 1:

```
exten => 123,1,Answer()
exten => 123,n,Background(main-menu)
exten => 123,n,WaitExten()

exten => 1,1,Playback(digits/1)

exten => 2,1,Playback(digits/2)

exten => 3,1,Playback(digits/3)

exten => 4,1,Playback(digits/4)
```

Наберите добавочный номер 123 и затем по подсказке главного меню введите 1. Почему Asterisk сразу же не воспроизводит этот номер? Потому, что цифра 1 неоднозначна; Asterisk не понимает, какой добавочный номер вызывается, 1 или 123. Он ожидает несколько секунд ввода другой цифры (например, 2 для вызова добавочного номера 123). Если набора никаких других цифр не последовало, по завершении времени ожидания Asterisk направляет вызов на добавочный номер 1. (Задавать собственные значения времени ожидания мы научимся в главе 6.)

Прежде чем двигаться дальше, посмотрим, что было сделано на данный момент. Вызвав наш диалплан, абоненты услышат приветствие. Если они нажмут 1, то услышат номер 1, если 2 – то номер 2 и т. д. Для начала это неплохо, но давайте это все немного усовершенствуем. С помощью приложения `Goto()` (Перейти к) заставим диалплан повторять приветствие после воспроизведения номера.

Как следует из его имени, *приложение* `Goto()` используется для перенаправления вызова в другую часть диалплана. Синтаксис `Goto()` требует передачи в него в качестве аргументов целевого контекста, добавочного номера и приоритета:

```
exten => 123,n,Goto(контекст,добавочныйномер,приоритет)
```

Теперь давайте применим приложение `Goto()` в нашем диалплане:

```
[incoming]
exten => 123,1,Answer()
exten => 123,n,Background(main-menu)

exten => 1,1,Playback(digits/1)
exten => 1,n,Goto(incoming,123,1)

exten => 2,1,Playback(digits/2)
exten => 2,n,Goto(incoming,123,1)
```

Две новые строки (выделенные курсивом) обеспечат возвращение управления над вызовом добавочному номеру 123 после воспроизведения выбранного номера.

Если вы внимательно посмотрите на приложение `Goto()`, то поймете, что в него на самом деле можно передавать один, два или три аргумента. Если передается только один аргумент, Asterisk предположит, что это основной приоритет текущего добавочного номера. Если передано два аргумента, Asterisk будет трактовать их как добавочный номер и приоритет, к которым надо перейти в текущем контексте.

В данном примере были переданы все три аргумента для наглядности, но, если бы мы задали только добавочный номер и приоритет, результат был бы аналогичным.

## Обработка ошибочных вводов и времени ожидания

Теперь, когда создание нашего первого голосового меню уже близится к завершению, введем специальные добавочные номера. Во-первых, нам необходим добавочный номер для недействительных вводов; когда абонент нажимает не ту кнопку (например, 9 для предыдущего примера), вызов направляется на добавочный номер `i`. Во-вторых, необходим добавочный номер для обработки ситуаций, когда абонент не производит ввод вовремя (время ожидания по умолчанию – 10 с). Если абонент слишком долго не нажимает кнопку после запуска приложения `WaitExten()`, вызовы направляются на добавочный номер `t`. Вот как будет выглядеть диалплан после введения этих двух добавочных номеров:

```
[incoming]
exten => 123,1,Answer()
exten => 123,n,Background(enter-ext-of-person)
exten => 123,n,WaitExten()
```

```
exten => 1,1,Playback(digits/1)
exten => 1,n,Goto(incoming,123,1)

exten => 2,1,Playback(digits/2)
exten => 2,n,Goto(incoming,123,1)

exten => 3,1,Playback(digits/3)
exten => 3,n,Goto(incoming,123,1)

exten => i,1,Playback(pbx-invalid)
exten => i,n,Goto(incoming,123,1)

exten => t,1,Playback(vm-goodbye)
exten => t,n,Hangup()
```

Использование добавочных номеров `i` и `t` делает диалплан несколько более надежным и практичным. Но надо сказать, он по-прежнему довольно примитивен, потому что внешние абоненты не имеют возможности соединения с реальным живым человеком. Для этого нам придется ознакомиться с еще одним приложением – `Dial()` (Звонить).

## Использование приложения Dial()

Одно из самых ценных свойств Asterisk – это возможность установления соединения между разными абонентами. Это особенно полезно, когда абоненты используют разные методы связи. Например, абонент A может звонить по традиционной аналоговой телефонной линии, тогда как пользователь B может сидеть в кафе в другой части света и говорить по IP-телефону. К счастью, большую часть тяжелой работы по установлению соединения и выполнению преобразований между разными сетями Asterisk берет на себя. От вас требуется лишь научиться использовать приложение `Dial()`.

Синтаксис `Dial()` немного сложнее, чем синтаксис приложений, которые применялись до сих пор, но не пугайтесь. `Dial()` принимает четыре аргумента. Первый – получатель вызова. Он состоит (в самой простой форме) из названия технологии (или транспортного протокола), с помощью которой выполняется вызов, символа слэш и имени удаленной конечной точки или ресурса. Самыми широко используемыми типами технологий являются Zap (для аналоговых каналов и каналов T1/E1/J1), SIP и IAX2. Например, допустим, требуется вызвать конечную точку Zap, определенную как `Zap/1`, которая представляет собой FXS-канал с подключенным к нему аналоговым телефоном. Технология – Zap, ресурс – 1. Аналогично, для вызова устройства SIP (описанного в sip.conf) получателем вызова может быть `SIP/Jane`, а для устройства IAX (описанного в iax.conf) – `IAX2/Fred`. Если бы потребовалось, чтобы при вызове добавочного номера 123 диалплана Asterisk звонил по каналу `Zap/1`, мы бы ввели следующий добавочный номер:

```
exten => 123,1,Dial(Zap/1)
```

Также можно звонить по нескольким каналам одновременно, объединяя получателей вызова с помощью символа амперсанда (&):

```
exten => 123,1,Dial(Zap/1&Zap/2&SIP/Jane)
```

Приложение Dial() будет дозваниваться всем заданным получателям вызовов одновременно и установит связь с любым из заданных каналов, который ответит первым. Если приложение не может связаться ни с одним вызываемым абонентом, Asterisk задаст переменной DIALSTATUS (статус звонка) значение, соответствующее ситуации невозможности дозвониться на вызываемые номера, и продолжит выполнение следующего приоритета добавочного номера[1].

Приложение Dial() также позволяет устанавливать связь с удаленной конечной точкой VoIP, которая не была предварительно описана в конфигурационных файлах канала. Вот полный синтаксис такого типа соединения:

```
Dial(технология/пользователь[:пароль]@удаленный_хост[:порт][/удаленный_добавочный_номер])
```

В качестве примера можно позвонить на демонстрационный сервер Digium, который использует протокол IAX2, по следующему добавочному номеру:

```
exten => 500,1,Dial(IAX2/guest@misery.digium.com/s)
```

Полный синтаксис приложения Dial() для звоноков по каналам Zap немного иной, как показано ниже:

```
Dial(Zap/[gGrR]канал_или_группа[/удаленный_добавочный_номер])
```

Например, вот как описывался бы вызов номера 1-800-555-1212 по Zap-каналу под номером 4.

```
exten => 501,1,Dial(Zap/4/18005551212)
```

Второй аргумент приложения Dial() – время ожидания, задаваемое в секундах. Если время ожидания задано, Dial() будет пытаться дозвониться по заданным номерам в течение этого количества секунд, а потом перейдет к следующему приоритету добавочного номера. Если время ожидания не задано, Dial() будет дозваниваться на вызываемые каналы до тех пор, пока кто-нибудь не ответит или пока вызывающий абонент не повесит трубку. Введем для нашего добавочного номера время ожидания 10 с:

```
exten => 123,1,Dial(Zap/1,10)
```

Если ответ на звонок получен до истечения времени ожидания, связь между каналами устанавливается и диалплан выполнен. Если вызываемый номер просто не отвечает, занят или недоступен по какой-то другой причине, Asterisk задаст переменную DIALSTATUS и перейдет к следующему приоритету добавочного номера.

---

[1] Не беспокойтесь, позже мы рассмотрим переменные (в разделе «Использование переменных») и покажем, как заставить диалплан принимать решения на основании значения переменной DIALSTATUS.

Давайте применим то, что изучили на данный момент, в другом примере:

```
exten => 123,1,Dial(Zap/1,10)
exten => 123,n,Playback(vm-nobodyavail)
exten => 123,n,Hangup()
```

Как видите, этот пример будет воспроизводить звуковой файл vm-nobodyavail.gsm в случае, если звонок остается без ответа.

Третий аргумент `Dial()` – строка опций. Она может содержать один или более символов, влияющих на поведение приложения `Dial()`. Список возможных опций слишком велик, чтобы приводить его здесь; рассмотрим лишь самую популярную из них – опцию `m`. Если указать `m` в качестве третьего аргумента, вызывающая сторона, пока выполняется дозвон до вызываемого абонента, будет слышать во время ожидания вместо гудков музыку (конечно, если эта музыка сконфигурирована правильно). Чтобы добавить опцию `m` в наш последний пример, просто изменим первую строку:

```
exten => 123,1,Dial(Zap/1,10,m)
exten => 123,n,Playback(vm-nobodyavail)
exten => 123,n,Hangup()
```

Теперь, когда мы научились использовать приложение `Dial()`, добавочные номера 1 и 2 в диалплане стали бесполезными. Давайте заменим их новыми добавочными номерами, которые позволят внешним абонентам дозваниваться до Джона (John) и Джейн (Jane):

```
[incoming]
exten => 123,1,Answer()
exten => 123,n,Background(enter-ext-of-person)
exten => 123,n,WaitExten()

exten => 1,1,Dial(Zap/1,10)
exten => 1,n,Playback(vm-nobodyavail)
exten => 1,n,Hangup()

exten => 2,1,Dial(SIP/Jane,10)
exten => 2,n,Playback(vm-nobodyavail)
exten => 2,n,Hangup()

exten => i,1,Playback(pbx-invalid)
exten => i,n,Goto(incoming,123,1)

exten => t,1,Playback(vm-goodbye)
exten => t,n,Hangup()
```

Четвертый и последний аргумент приложения `Dial()` – URL. Если вызываемый канал поддерживает прием URL в момент вызова, заданный URL будет передан (например, если используется IP-телефон, поддерживающий прием URL, этот URL появится на дисплее телефона; аналогично, если используется программный телефон, URL может быть выведен на экран монитора). Этот аргумент применяется очень редко.

Обратите внимание, что второй, третий и четвертый аргументы могут быть опущены. Например, если требуется определить опцию, но при этом вы не собираетесь задавать время ожидания, просто оставьте пропуск на месте аргумента времени ожидания, как в данном примере:

```
exten => 1,1,Dial(Zap/1,,m)
```

## Добавление контекста для внутренних вызовов

До сих пор в наших примерах мы ограничивались одним контекстом, но, вероятно, справедливо предполагать, что в диалпланах практически всех установок Asterisk будет не один контекст, а больше. Как упоминалось в начале данной главы, одна из важных функций контекстов – разделение прав доступа (таких, как осуществление междугородних вызовов или звонков на определенные добавочные номера) для разных классов абонентов. В следующем примере наш диалплан будет дополнен созданием двух внутренних добавочных номеров, для которых будет настроена возможность звонить друг другу. Для этого создадим новый контекст, `[employees]` (служащие).

Как и в предыдущих примерах, предполагаем, что аналоговый канал FXS (`Zap/1` в данном случае) уже сконфигурирован и что настройки файла zapata.conf таковы, что все вызовы, берущие начало в `Zap/1`, обрабатываются в контексте `[employees]`. В нескольких примерах в конце главы также будет предполагаться, что Zap-канал FXO сконфигурирован как `Zap/4` и вызовы, поступающие на этот канал, направляются в контекст `[incoming]`.

Также мы предположили, что имеется по крайней мере один SIP-канал (названный `SIP/Jane`), который используется в контексте `[employees]`. Это было сделано, чтобы показать пример использования других типов каналов.

Если вы не располагаете оборудованием для организации перечисленных выше каналов (таких, как `Zap/4`) или если используете другие имена каналов (например, не `SIP/Jane`), просто скорректируйте примеры соответственно конфигурации своей системы.

Теперь наш диалплан выглядит так:

```
[incoming]
exten => 123,1,Answer()
exten => 123,n,Background(enter-ext-of-person)
exten => 123,n,WaitExten()

exten => 1,1,Dial(Zap/1,10)
exten => 1,n,Playback(vm-nobodyavail)
exten => 1,n,Hangup()

exten => 2,1,Dial(SIP/Jane,10)
exten => 2,n,Playback(vm-nobodyavail)
exten => 2,n,Hangup()
```

```
exten => i,1,Playback(pbx-invalid)
exten => i,n,Goto(incoming,123,1)

exten => t,1,Playback(vm-goodbye)
exten => t,n,Hangup()

[employees]
exten => 101,1,Dial(Zap/1)

exten => 102,1,Dial(SIP/Jane)
```

В этом примере в контекст `[employees]` было добавлено два новых добавочных номера. Таким образом, человек, использующий канал `Zap/1`, может поднять трубку телефона и позвонить человеку, находящемуся на линии `SIP/Jane`, набрав номер 102. Точно так же телефон, зарегистрированный как `SIP/Jane`, может позвонить `Zap/1`, если пользователь наберет 101.

Добавочные номера 101 и 102 выбраны для примера произвольно, для своих добавочных номеров вы можете использовать любые другие цифры. Также необходимо помнить, что вы не ограничены трехзначными добавочными номерами; номер может включать столько угодно цифр. (Скажем так, почти сколько угодно. Добавочные номера не должны быть длиннее 80 символов, и нельзя использовать добавочные номера длиной в один символ для собственных нужд, поскольку они зарезервированы.) Не забывайте, что могут применяться и имена, как в данном примере:

```
[incoming]
exten => 123,1,Answer()
exten => 123,n,Background(enter-ext-of-person)
exten => 123,n,WaitExten()

exten => 1,1,Dial(Zap/1,10)
exten => 1,n,Playback(vm-nobodyavail)
exten => 1,n,Hangup()

exten => 2,1,Dial(SIP/Jane,10)
exten => 2,n,Playback(vm-nobodyavail)
exten => 2,n,Hangup()

exten => i,1,Playback(pbx-invalid)
exten => i,n,Goto(incoming,123,1)

exten => t,1,Playback(vm-goodbye)
exten => t,n,Hangup()

[employees]
exten => 101,1,Dial(Zap/1)
exten => john,1,Dial(Zap/1)

exten => 102,1,Dial(SIP/Jane)
exten => jane,1,Dial(SIP/Jane)
```

Конечно, не помешало бы добавить именные добавочные номера, если предполагается, что пользователи могут получать звонки по VoIP-протоколу, такому как SIP, который поддерживает вызов по имени. Также нетрудно заметить, что в диалплане могут быть разные добавочные номера для вызова одной конечной точки, например добавочный номер 200 с выходом на канал `SIP/George` и добавочный номер 201, который воспроизводит некоторое сообщение, а *затем* звонит `SIP/George`.

Теперь, когда наши внутренние абоненты могут звонить друг другу, мы значительно продвинулись на пути к созданию полного диалплана. Далее будет показано, как можно сделать диалплан более масштабируемым и пригодным к внесению изменений в будущем.

## Использование переменных

Переменные, используемые в диалплане Asterisk, способствуют сокращению объема вводимого текста, делают код более понятным или вводят дополнительную логику. Тем, кто имеет опыт разработки программного обеспечения, вероятно, понятие переменной уже знакомо. Если нет, не стоит беспокоиться; мы объясним, что такое переменные и как они используются.

Переменные можно рассматривать как контейнер, в котором в данный момент времени может храниться одно значение. Например, мы могли бы создать переменную `JOHN` и присвоить ей значение `Zap/1`. Теперь при написании диалплана можно ссылаться на канал Джона по имени, а не запоминать, что Джон использует канал, названный `Zap/1`.

Существует два способа использования переменной. Чтобы сослаться на имя переменной, просто вводится ее имя, например `JOHN`. Если, с другой стороны, требуется сослаться на ее значение, необходимо ввести знак доллара, открывающую фигурную скобку, имя переменной и закрывающую фигурную скобку. Вот как используется переменная в приложении `Dial()`:

```
exten => 555,1,Dial(${JOHN})
```

В нашем диалплане Asterisk будет автоматически заменять все ссылки `${JOHN}` значением, присвоенным переменной под именем `JOHN`.

Обратите внимание, что имена переменных чувствительны к регистру. `JOHN` и `John` – это разные переменные. Для удобства чтения все имена переменных в примерах будут записываться в верхнем регистре. Также следует помнить, что все переменные, заданные Asterisk, тоже будут записаны прописными буквами. Некоторые переменные, такие как `CHANNEL` или `EXTEN`, зарезервированы Asterisk. Не надо пытаться задавать их.

В диалплане используется три типа переменных: глобальные переменные, переменные канала и переменные среды. Кратко рассмотрим каждый из этих типов.

## Глобальные переменные

Как следует из их названия, *глобальные* переменные применяются ко всем добавочным номерам во всех контекстах. Глобальные переменные полезны тем, что могут использоваться в любом месте диалплана, повышая читабельность и обслуживаемость кода. Предположим, имеется большой диалплан и несколько сотен ссылок на канал `Zap/1`. Теперь представим, что необходимо пересмотреть весь диалплан и изменить все эти ссылки на `Zap/2`. Это был бы, мягко выражаясь, долгий и чреватый ошибками процесс.

Но если бы в начале диалплана была определена переменная со значением `Zap/1` и далее использовались лишь ссылки на нее, потребовалось бы изменить только одну строку.

Глобальные переменные объявляются в контексте `[globals]` в начале файла extensions.conf. Их можно также задать программно с помощью функции диалплана `GLOBAL()`[1]. Вот пример использования обоих методов задания переменных в диалплане. В первом варианте глобальной переменной `JOHN` присваивается значение `Zap/1`. Эта переменная задается в момент, когда Asterisk выполняет синтаксический разбор диалплана. Второй пример представляет, как можно задать глобальную переменную в процессе выполнения диалплана. В этом случае переменной `George` присваивается значение `SIP/George` при выполнении звонка на добавочный номер 124 в контексте `[employees]`:

```
[globals]
JOHN=Zap/1

[employees]
exten => 124,1,Set(GLOBAL(GEORGE)=SIP/George)
```

## Переменные канала

Переменная *канала* – это переменная, связанная только с конкретным вызовом. В отличие от глобальных переменных, переменные каналов определяются только на время текущего вызова и доступны лишь для каналов, участвующих в нем.

Для использования в диалплане предопределено множество переменных каналов. Они описаны в файле channelvariables.txt, находящемся в подпапке doc папки исходного кода Asterisk. Переменные каналов задаются с помощью приложения `Set()`:

```
exten => 125,1,Set(MAGICNUMBER=42)
```

Многие варианты использования переменных каналов будут рассмотрены в главе 6.

---

[1] Не беспокойтесь! Функции диалплана будут рассмотрены в разделе «Функции диалплана» главы 6.

### Переменные среды

Переменные *среды* – это средство организации доступа к переменным среды UNIX из Asterisk. Для их использования служит функция диалплана `ENV()`. Ее синтаксис выглядит следующим образом: `${ENV(var)}`, где *var* – переменная среды UNIX, на которую выполняется ссылка. Переменные среды используются в диалпланах Asterisk не часто, но они доступны на случай необходимости.

### Добавление переменных в диалплан

Теперь, ознакомившись с переменными, применим их в нашем диалплане. Добавим глобальные переменные для двух людей, Джона и Джейн:

```
[globals]
JOHN=Zap/1
JANE=SIP/Jane

[incoming]
exten => 123,1,Answer()
exten => 123,n,Background(enter-ext-of-person)
exten => 123,n,WaitExten()

exten => 1,1,Dial(${JOHN},10)
exten => 1,n,Playback(vm-nobodyavail)
exten => 1,n,Hangup()

exten => 2,1,Dial(${JANE},10)
exten => 2,n,Playback(vm-nobodyavail)
exten => 2,n,Hangup()

exten => i,1,Playback(pbx-invalid)
exten => i,n,Goto(incoming,123,1)

exten => t,1,Playback(vm-goodbye)
exten => t,n,Hangup()

[employees]

exten => 101,1,Dial(${JOHN})
exten => john,1,Dial(${JOHN})

exten => 102,1,Dial(${JANE})
exten => jane,1,Dial(${JANE})
```

## Сопоставление с шаблонами

Если мы хотим предоставить людям возможность осуществлять звонки *через* Asterisk и желаем, чтобы Asterisk обеспечивала соединение абонента с внешним ресурсом, нам необходим механизм сопоставления любого телефонного номера, который может быть набран абонентом. Можете себе представить, как утомительно было бы вручную пи-

сать диалплан с добавочными номерами для всех возможных вариантов? К счастью, у Asterisk есть как раз то, что надо для таких случаев: *сопоставление с шаблонами*. Благодаря возможности сопоставления с шаблонами в диалплане можно создать один добавочный номер, который будет соответствовать множеству разных номеров.

## Синтаксис сопоставления с шаблонами

Используемые в шаблонах буквы и символы представляют определенные группы символов. Шаблоны всегда начинаются с символа подчеркивания (_). Он указывает Asterisk, что выполняется сопоставление с шаблоном, а не с явно заданным добавочным номером. (Безусловно, это означает, что имена добавочных номеров нельзя начинать с символа подчеркивания.)

Если не поставить символ подчеркивания в начале шаблона, Asterisk посчитает, что это просто именованный добавочный номер, и не будет выполнять сопоставления с шаблоном. Это одна из самых распространенных ошибок среди новичков в Asterisk.

После подчерка может использоваться один или более символов из перечисленных ниже.

`X`

Соответствует любому одиночному числу от 0 до 9.

`Z`

Соответствует любому одиночному числу от 1 до 9.

`N`

Соответствует любому одиночному числу от 2 до 9.

`[15-7]`

Соответствует любому однозначному числу из заданного диапазона. В данном случае шаблон соответствует одиночной цифре 1, 5, 6 или 7.

`.` (точка)

Универсальное соответствие; соответствует *одному или более* символам, любым.

Если не быть осторожным, сопоставления с групповым символом могут привести к тому, что диалплан будет делать совсем не то, что предполагается (например, сопоставление с встроенными добавочными номерами, такими как `i` или `h`). Универсальное соответствие должно использоваться в шаблоне только после того, как сопоставлено максимально возможное количество цифр. Например, следующий шаблон, наверное, не должен применяться никогда:

```
_.
```

На самом деле Asterisk предупредит в случае попытки его применения. Лучше по возможности используйте такой шаблон:

```
_X.
```

! (восклицательный знак)

Универсальное соответствие; соответствует *нулю или более* символам, любым.

Чтобы использовать сопоставление с шаблонами в своем диалплане, просто вставьте шаблон на место добавочного номера (или его имени):

```
exten => _NXX,1,Playback(auth-thankyou)
```

В этом примере шаблон соответствует трехзначному добавочному номеру в диапазоне от 200 до 999 (N соответствует любой цифре от 2 до 9, а каждый X – от 0 до 9). То есть, если бы абонент набрал любой трехзначный добавочный номер в диапазоне от 200 до 999 в данном контексте, он бы услышал звуковой файл auth-thankyou.gsm.

Еще одна важная деталь, которую необходимо знать о сопоставлении с шаблонами: если Asterisk находит более одного шаблона, соответствующего набранному добавочному номеру, она будет использовать *наиболее точный* из них (слева направо). Скажем, задано два следующих шаблона и абонент набирает 555-1212:

```
exten => _555XXXX,1,Playback(digits/1)
exten => _55512XX,1,Playback(digits/2)
```

В данном случае был бы выбран второй добавочный номер, потому что он более точно соответствует набранному номеру.

## Примеры сопоставления с шаблонами

Прежде чем продолжить, рассмотрим еще несколько примеров сопоставления с шаблонами. В каждом из них проверьте, сможете ли вы сказать, чему соответствует шаблон, до того, как прочитаете объяснения. Начнем с простого:

```
_NXXXXXX
```

Этот шаблон соответствует любому семизначному номеру, начинающемуся с двойки и выше, то есть любому локальному семизначному номеру по североамериканскому плану нумерации. В зонах, где используются 10-значные номера, этот шаблон выглядел бы так:

```
_NXXNXXXXXX
```

### NANP и мошенничество с оплатой звонков

Североамериканский план нумерации (North American Numbering Plan, NANP) – это общая схема телефонной нумерации, используемая 19 странами в Северной Америке и Канаде. Код страны для стран NANP – 1.

В США и Канаде нормы и правила электросвязи довольно похожи (и удобны), поэтому на большинство междугородных номеров можно звонить, используя код страны 1, по разумной цене.

Однако многие даже не догадываются, что NANP используют 19 стран, подчас с очень разными правилами электросвязи. (Более подробную информацию об этом можно найти по адресу *http://www.nanpa.com*.)

Очень популярно мошенничество с использованием NANP, когда наивных американцев обманным путем заставляют звонить в страны Карибского бассейна по номерам, звонок по которым оплачивается поминутно, как междугородний; абоненты верят, что, если набрать 1-NPA-NXX-XXXX, звонок будет оплачиваться по стандартным национальным тарифам на междугородние звонки. Поскольку в данном государстве могут действовать правила, допускающие такого рода мошенничество, абоненту в итоге приходится оплачивать телефонные счета.

Единственный способ предотвратить такого рода деятельность – блокировать определенные междугородние коды (например, 809) и снимать ограничения только в случае необходимости.

Обратите внимание, что ни один из данных шаблонов не станет обрабатывать междугородние звонки. Скоро мы рассмотрим этот вопрос.

Попробуем другой шаблон:

```
_1NXXNXXXXXX
```

Данный шаблон немного сложнее. Он соответствует 1, за которой следует код города от 200 до 999, а затем – любой семизначный номер. В зоне действия NANP этот шаблон будет использоваться для сопоставления с любым междугородним номером[1].

Теперь еще более хитрый пример:

```
_011.
```

Если, увидев этот шаблон, вы лишь почесали затылок, посмотрите на него еще раз. Заметили точку в конце? Этот шаблон соответствует любому номеру, начинающемуся с 011 и имеющему по крайней мере еще одну цифру. В NANP это соответствует международному телефонному номеру. (Мы будем использовать эти шаблоны в следующем разделе для добавления в наш диалплан возможностей выполнения исходящих звонков.)

[1] Те, кто вырос в Северной Америке, возможно, думают, что 1, которую они набирают при звонках по межгороду, – это «международный код». Это неправильно. 1 – это международный код страны для всех стран, использующих NANP. Помните это, если вам когда-нибудь придется давать свой номер телефона кому-то из другой страны. Они могут не знать кода вашей страны и, таким образом, не смогут дозвониться вам, имея только код города и номер телефона. Полный номер телефона с кодом страны записывается так: +1 NPA NXX XXXX (где NPA – код города) – например +1 416 555 1212.

## Использование переменной канала ${EXTEN}

Мы знаем, о чем вы думаете... Вы сидите и задаете себе вопрос: «Так что же делать, если я хочу использовать сопоставление с шаблонами, но мне надо знать, какие цифры набираются?» К счастью, у Asterisk есть ответ. При каждом звонке на добавочный номер Asterisk сохраняет набранный номер в переменной канала `${EXTEN}`. Чтобы протестировать ее, можно использовать приложение `SayDigits()`:

```
exten => _XXX,1,SayDigits(${EXTEN})
```

В этом примере приложение `SayDigits()` будет воспроизводить для вас набранный вами трехзначный добавочный номер.

Часто переменную `${EXTEN}` удобно использовать для удаления определенного количества цифр в начале добавочного номера. Это осуществляется с помощью синтаксиса `${EXTEN:`*x*`}`, где *x* – разряд, с которого должна начинаться возвращаемая строка, считая слева направо. Например, если значение `EXTEN` – `95551212`, `${EXTEN:1}` равно `5551212`. Рассмотрим другой пример:

```
exten => _XXX,1,SayDigits(${EXTEN:1})
```

В этом примере приложение `SayDigits()` начнет воспроизведение со второй цифры, таким образом, будут воспроизведены только две последние цифры набранного добавочного номера.

### Более сложные операции с номерами

Полный синтаксис переменной `${EXTEN}` – `${EXTEN:`*x*`:`*y*`}`, где *x* – начальное положение, а *y* – количество цифр, которое должно быть возвращено. Пусть задана строка набора:

```
94169671111
```

Используя конструкцию `${EXTEN:`*x*`:`*y*`}`, можно извлечь следующие цифровые строки:

`${EXTEN:1:3}` – будет возвращена строка `416`.

`${EXTEN:4:7}` – будет возвращена строка `9671111`.

`${EXTEN:-4:4}` – строка будет начинаться с четвертого символа с конца и включает четыре символа, что даст `1111`.

`${EXTEN:1}` – будет возвращено все после первой цифры, `4169671111` (если количество цифр, которое должно быть возвращено, не задано, будет возвращена вся оставшаяся строка).

Это очень мощная структура, но большинство из приведенных ее вариантов используются редко. Чаще всего вы будете применять `${EXTEN:1}`, чтобы отбросить внешний код доступа.

## Активация исходящих вызовов

Теперь, когда мы ознакомились с шаблонами, можно переходить к тому, как обеспечить абонентам возможность осуществлять исходящие звонки. Первое, что мы сделаем, – добавим переменную в контекст `[globals]`, чтобы определить, какой канал будет использоваться для исходящих вызовов:

```
[globals]
JOHN=Zap/1
JANE=SIP/Jane
OUTBOUNDTRUNK=Zap/4
```

Далее добавим в диалплан контекст для исходящих вызовов.

Возможно, сейчас вы задаетесь вопросом: «Зачем нужен отдельный контекст для исходящих звонков?» Это делается для того, чтобы иметь возможность управлять тем, какие абоненты имеют право делать исходящие звонки и какого типа могут быть эти звонки.

Сначала создадим контекст для локальных вызовов. Не будем отступать от традиций и первой цифрой в наших шаблонах поставим 9, чтобы пользователи для звонка на внешний номер набирали 9:

```
[outbound-local]
exten => _9NXXXXXX,1,Dial(${OUTBOUNDTRUNK}/${EXTEN:1})
exten => _9NXXXXXX,n,Congestion()
exten => _9NXXXXXX,n,Hangup()
```

Обратите внимание, что цифра 9 на самом деле не обеспечивает выхода на внешнюю линию, как это происходит во многих традиционных системах офисных АТС. После набора цифры 9 в аналоговой линии сразу же пропадет тональный сигнал готовности линии. Если вам хочется, чтобы зуммер продолжался даже после набора цифры 9, добавьте следующую строку (сразу после описания контекста):

```
ignorepat => 9
```

Согласно данной директиве Asterisk будет продолжать давать тональный сигнал в аналоговую линию даже после набора указанного шаблона. Это не будет работать с телефонами VoIP, поскольку обычно они не передают отдельные цифры номера в систему во время их ввода; они отправляют Asterisk весь номер сразу. К счастью, большинство популярных VoIP-телефонов можно настроить на имитацию такой функциональности.

Повторим, что было сделано. Мы добавили глобальную переменную `OUTBOUNDTRUNK`, которая просто определяет канал, используемый для исходящих вызовов[1]. Также был введен контекст для локальных исходя-

[1] Это обеспечивает то преимущество, что, если однажды будет принято решение передавать вызовы по какому-то другому каналу, надо будет отредактировать имя канала, заданное как значение переменной `OUTBOUNDTRUNK` только в контексте `[globals]`, а не менять вручную все ссылки на этот канал по всему диалплану.

щих вызовов. В приоритете 1 с помощью синтаксиса ${EXTEN:1} от набранного добавочного номера отбрасывается цифра 9 и делается попытка дозвониться на этот номер по каналу, указанному переменной OUTBOUNDTRUNK. Если удается дозвониться, абонент соединяется с исходящим каналом. Если вызов заканчивается неудачей (например, канал занят или невозможно набрать номер по какой-то другой причине), вызывается приложение Congestion() (перегрузка), которое воспроизводит «короткие гудки» (тональный сигнал перегрузки линии), сообщая абоненту, что дозвониться не удалось.

Прежде чем двигаться дальше, убедимся, что наш диалплан позволяет выполнять исходящие звонки на номера экстренного вызова:

```
[outbound-local]
exten => _9NXXXXXX,1,Dial(${OUTBOUNDTRUNK}/${EXTEN:1})
exten => _9NXXXXXX,n,Congestion()
exten => _9NXXXXXX,n,Hangup()

exten => 911,1,Dial(${OUTBOUNDTRUNK}/911)
exten => 9911,1,Dial(${OUTBOUNDTRUNK}/911) ; Чтобы тот, кто набрал первую "9",
                                           ; тоже мог дозвониться
```

Опять же, в данном примере предполагается, что мы находимся в США или Канаде. Если вы проживаете в другой стране, замените, пожалуйста, 911 номером экстренных служб, используемым в вашем регионе. Это то, о чем ни в коем случае нельзя забывать при создании своего диалплана!

Далее добавим контекст для междугородних звонков:

```
[outbound-long-distance]
exten => _91NXXNXXXXXX,1,Dial(${OUTBOUNDTRUNK}/${EXTEN:1})
exten => _91NXXNXXXXXX,n,Playtones(congestion)
exten => _91NXXNXXXXXX,n,Hangup()
```

Теперь, когда у нас есть два новых контекста, как предоставить возможность внутренним абонентам их использовать? Необходим какой-то способ использовать функциональность одного контекста из другого.

## Выражения include

Asterisk предоставляет возможность использовать добавочные номера из одного контекста в другом контексте с помощью директивы include (включить). Так можно управлять доступом к различным разделам диалплана. Мы будем применять функциональность включения, чтобы дать возможность пользователям из контекста [employees] делать исходящие звонки. Но сначала давайте рассмотрим синтаксис.

Выражение include имеет следующий вид, где *контекст* – имя удаленного контекста, который требуется включить в текущий:

```
include => контекст
```

При включении контекстов друг в друга необходимо внимательно продумать порядок их включения. Asterisk сначала будет пытаться найти

соответствие набранному добавочному номеру в текущем контексте. В случае неудачи она будет рассматривать контекст, включенный первым (в том числе все включенные в него контексты), а затем будет переходить от одного контекста к другому в порядке их включения.

На данный момент в нашем диалплане есть два контекста для исходящих вызовов, но абоненты из контекста `[employees]` не могут их использовать. Исправим это, включив оба исходящих контекста в контекст `[employees]`, как показано в примере:

```
[globals]
JOHN=Zap/1
JANE=SIP/Jane
OUTBOUNDTRUNK=Zap/4

[incoming]
exten => 123,1,Answer()
exten => 123,n,Background(enter-ext-of-person)
exten => 123,n,WaitExten()

exten => 1,1,Dial(${JOHN},10)
exten => 1,n,Playback(vm-nobodyavail)
exten => 1,n,Hangup()

exten => 2,1,Dial(${JANE},10)
exten => 2,n,Playback(vm-nobodyavail)
exten => 2,n,Hangup()

exten => i,1,Playback(pbx-invalid)
exten => i,n,Goto(incoming,123,1)

exten => t,1,Playback(vm-goodbye)
exten => t,n,Hangup()

[employees]
include => outbound-local
include => outbound-long-distance

exten => 101,1,Dial(${JOHN})
exten => john,1,Dial(${JOHN})
exten => 102,1,Dial(${JANE})
exten => jane,1,Dial(${JANE})

[outbound-local]
exten => _9NXXXXXX,1,Dial(${OUTBOUNDTRUNK}/${EXTEN:1})
exten => _9NXXXXXX,n,Congestion()
exten => _9NXXXXXX,n,Hangup()

exten => 911,1,Dial(${OUTBOUNDTRUNK}/911)
exten => 9911,1,Dial(${OUTBOUNDTRUNK}/911)

[outbound-long-distance]
exten => _91NXXNXXXXXX,1,Dial(${OUTBOUNDTRUNK}/${EXTEN:1})
```

```
exten => _91NXXNXXXXXX,n,Playtones(congestion)
exten => _91NXXNXXXXXX,n,Hangup()
```

Эти два выражения `include` обеспечивают абонентам из контекста `[employees]` возможность осуществлять исходящие вызовы. Также нужно заметить следующее: в целях безопасности всегда необходимо убедиться в том, что контекст `[inbound]` (входящие) не допускает возможности исходящих звонков. (Если вдруг это стало бы возможным, люди могли бы дозваниваться в вашу систему, а затем делать исходящие платные звонки за ваш счет!)

## Заключение

И вот он готов – базовый, но вполне функциональный диалплан. Это не совсем полный набор возможных свойств, но основные из них были рассмотрены. В следующих главах наш базовый диалплан будет дополнен новыми функциями.

Если какие-то части данного диалплана вызывают у вас вопросы, вероятно, вам следует вернуться немного назад и перечитать раздел или два, прежде чем переходить к следующей главе. Крайне важно понимать все эти основные принципы и их применение, потому что они являются основой для изучения следующих глав.

# 6

# Дополнительные концепции диалплана

*Чтобы получить список всех направлений,*
*в которых технологии не смогли улучшить*
*качество жизни, пожалуйста, нажмите три.*
– Элис Кан

Отлично. Основные принципы диалплана рассмотрены, но многое впереди. Если вы еще не до конца разобрались с предыдущей главой, пожалуйста, вернитесь назад и перечитайте ее. Мы собираемся переходить к более сложным вопросам.

## Выражения и работа с переменными

Поскольку мы начинаем погружение в более глубокие аспекты диалплана, пришло время представить несколько инструментов, которые могут значительно увеличить его мощь. Эти структуры невероятно усложнят логику диалплана, обеспечив ему возможность принимать решения на основании различных задаваемых критериев. Итак, собрались с мыслями – и начнем.

### Элементарные выражения

Выражения – это сочетания переменных, операторов и значений, объединяемые для получения результата. Выражение может тестировать значения, изменять строки или выполнять вычисления. Допустим, имеется переменная `COUNT`. Возьмем два возможных выражения, составленных с использованием этой переменной: «`COUNT` плюс 1» и «`COUNT` разделить на 2». Каждое из этих выражений имеет конкретный результат или значение, зависящие от значения данной переменной.

В Asterisk выражения всегда начинаются со знака доллара ($) и открывающей квадратной скобки и заканчиваются закрывающей квадратной скобкой:

```
$[выражение]
```

Таким образом, упомянутые выше примеры будут записаны так:

```
$[${COUNT} + 1]
$[${COUNT} / 2]
```

Когда Asterisk встречает в диалплане выражение, она заменяет его результирующим значением. Важно отметить, что это происходит *после* подстановки переменных. Чтобы продемонстрировать это, посмотрим на следующий фрагмент кода[1]:

```
exten => 321,1,Set(COUNT=3)
exten => 321,n,Set(NEWCOUNT=$[${COUNT} + 1])
exten => 321,n,SayNumber(${NEWCOUNT})
```

В первом приоритете переменной COUNT присваивается значение 3.

Во втором приоритете участвует только одно приложение, Set(), но фактически выполняется три действия:

1. Asterisk подставляет в выражении вместо ${COUNT} число 3. Выражение, в сущности, превращается в следующее:

   ```
   exten => 321,n,Set(NEWCOUNT=$[3 + 1])
   ```

2. Asterisk вычисляет выражение, суммируя 1 и 3, и заменяет его вычисленным значением, 4:

   ```
   exten => 321,n,Set(NEWCOUNT=4)
   ```

3. Приложение Set() присваивает значение 4 переменной NEWCOUNT.

Третий приоритет просто вызывает приложение SayNumber(), которое воспроизводит текущее значение переменной ${NEWCOUNT} (ей было присвоено значение 4 во втором приоритете).

Попробуйте это в своем диалплане.

## Операторы

Код диалплана Asterisk записывается на специальном языке сценариев. Это означает, что язык диалплана Asterisk, как любой язык программирования, распознает символы, называемые *операторами*, которые позволяют оперировать переменными. Рассмотрим типы операторов, используемые в Asterisk:

*Логические (булевы) операторы*

Эти операторы оценивают истинность выражения. С точки зрения вычислений это, по сути, означает, является ли выражение чем-то или оно является ничем (отличный от нуля или нуль, истина или ложь, включен или выключен и т. д.). К логическим операторам относятся:

[1] Помните, что при ссылке на переменную используется просто ее имя, но при ссылке на значение переменной ее имя должно быть заключено в квадратные скобки и перед ними должен стоять знак доллара.

*expr1 | expr2*

Этот оператор (называемый оператором ИЛИ) в случае истинности выражения *expr1* (не пустая строка и не нуль) возвращает результат его вычисления. В противном случае он возвращает результат вычисления выражения *expr2*.

*expr1 & expr2*

Это оператор (называемый оператором И) возвращает результат вычисления *expr1*, если оба выражения истинны (то есть если ни одно из выражений не дает в результате пустой строки или нуля). В противном случае возвращается нуль.

*expr1 {=, >, >=, <, <=, !=} expr2*

Эти операторы возвращают результаты сравнения целых чисел, если оба аргумента являются целыми числами; в противном случае возвращаются результаты сравнения строк. В результате сравнения получаем 1, если заданное отношение выполняется, или 0, если отношение не выполняется. (Сравнение строк выполняется соответственно текущим локальным настройкам операционной системы.)

*Арифметические операторы*

Хотите выполнить вычисление? Вам потребуется один из следующих операторов:

*expr1 {+, -} expr2*

Эти операторы возвращают результаты сложения или вычитания целочисленных аргументов.

*expr1 {*, /, %} expr2*

Эти операторы возвращают результаты умножения, целочисленного деления или остаток от деления целочисленных аргументов соответственно.

*Оператор регулярного выражения*

Также в Asterisk может использоваться оператор регулярного выражения:

*expr1 : expr2*

Этот оператор сравнивает выражение *expr1* с *expr2*, где последнее должно быть регулярным выражением[1]. Регулярное выражение привязывается к началу строки посредством явного задания ^[2].

---

[1] Больше информации о регулярных выражениях можно найти в полном справочнике Джеффри Е. Ф. Фридла (Jeffrey E. F. Friedl) «Mastering Regular Expressions» (издательство O'Reilly) или по адресу *http://www.regular-expressions.info*.

[2] Если вы не знаете, что делает ^ с регулярными выражениями, вы просто обязаны достать экземпляр книги «Mastering Regular Expressions». Она изменит вашу жизнь!

Если соответствие установлено и шаблон содержит по крайней мере одну подстроку регулярного выражения, `\( ... \)`, возвращается строка, соответствующая `\1`; в противном случае оператор сопоставления возвращает число совпавших символов. Если соответствия не выявлено и шаблон не содержит подстроку регулярного выражения, возвращается нулевая строка; в противном случае, возвращается `0`.

Синтаксический анализатор Asterisk версии 1.0 был довольно прост, поэтому обязательным требованием было отделение операторов от всех остальных значений как минимум одним пробелом. Соответственно, следующая запись не обеспечивала бы желаемого результата:

```
exten => 123,1,Set(TEST=$[2+1])
```

Такая запись привела бы к присвоению переменной `TEST` строки `2+1`, а не значения 3. Чтобы исправить это, надо поставить пробелы перед оператором и после него:

```
exten => 234,1,Set(TEST=$[2 + 1])
```

В Asterisk 1.2 или 1.4 это уже не является обязательным требованием, потому что синтаксический анализатор выражений был доработан с учетом таких типов сценариев. Однако ради удобства чтения по-прежнему рекомендуется отделять операторы пробелами.

Чтобы добавить текст в начало или конец переменной, просто поместите его в выражении рядом:

```
exten => 234,1,Set(NEWTEST=$[blah${TEST}])
```

# Функции диалплана

Функции диалплана делают выражения более мощными; их можно рассматривать как интеллектуальные переменные. Функций диалплана позволяют вычислять длины строк, даты и время, контрольные суммы MD5 и т. д., и все в рамках выражения диалплана.

## Синтаксис

Функции диалплана имеют следующий основной синтаксис:

```
ИМЯ_ФУНКЦИИ(аргумент)
```

Как и для переменных, ссылка на *имя* функции выполняется, как показано выше, а чтобы использовать *значение* функции, необходимо поставить впереди знак доллара и заключить функцию в фигурные скобки:

```
${ИМЯ_ФУНКЦИИ(аргумент)}
```

Также функции могут инкапсулировать (содержать в себе) другие функции:

```
${ИМЯ_ФУНКЦИИ(${ИМЯ_ФУНКЦИИ(аргумент)})}
```

```
1           2 3           4         4321
```

Как вы, вероятно, уже заметили, необходимо быть очень внимательным с открывающими и закрывающими скобками. В примере выше парные открывающие и закрывающие круглые и фигурные скобки обозначены одинаковыми числами.

## Примеры функций диалплана

Часто функции используются в сочетании с приложением `Set()` для получения или задания значения переменной. В качестве простого примера рассмотрим функцию `LEN()`. Эта функция вычисляет длину строки, заданной в качестве ее аргумента. Вычислим длину строки переменной и воспроизведем это значение для абонента:

```
exten => 123,1,Set(TEST=example)
exten => 123,n,SayNumber(${LEN(${TEST})})
```

Приведенный пример определит, что строка `example` содержит семь символов, задаст это значение как длину переменной и воспроизведет это число пользователю с помощью приложения `SayNumber()`.

Рассмотрим еще один простой пример. Если бы мы захотели задать время ожидания для одного из каналов, то могли бы использовать функцию `TIMEOUT()`. Функция `TIMEOUT()` принимает один из трех аргументов: `absolute` (абсолютное), `digit` (между цифрами) и `response` (ответ). Чтобы задать максимальный промежуток времени между вводом цифр с помощью функции `TIMEOUT()`, можно воспользоваться приложением `Set()`:

```
exten => s,1,Set(TIMEOUT(digit)=30)
```

Обратите внимание на то, что функция не заключена в символы `${ }`. Как и присвоение значения переменной, присвоение значения функции выполняется без использования символов `${ }`.

Полный список доступных функций можно получить, введя команду `core show functions` в интерфейсе командной строки Asterisk. Также они представлены в приложении F.

# Выполнение переходов по условию

После краткого ознакомления с выражениями и функциями пришло время использовать их на практике. Применение выражений и функций позволяет усложнить логику диалплана. Возможность принятия решений в диалплане обеспечивается выполнением *переходов по условию*. Остановимся на этом подробнее.

## Приложение GotoIf()

Ключ к выполнению переходов по условию — приложение `GotoIf()`. `GotoIf()` вычисляет выражение и отправляет абонента в соответствующее место назначения в зависимости от истинности или ложности выражения.

GotoIf() использует особый синтаксис, который часто называют условным:

```
GotoIf(выражение?местоназначения1:местоназначения2)
```

Если выражение истинно (возвращает значение true), абонент направляется на *местоназначения1*. Если выражение ложно (возвращает значение false), абонент направляется по второму адресу. Итак, что обеспечивает возвращение значения true или false? Пустая строка и номер 0 обеспечивают false, все остальное – true.

В качестве места назначения может быть задано следующее:

- Метка приоритета в рамках того же добавочного номера, например weasels.
- Добавочный номер и метка приоритета в рамках того же контекста, например 123,weasels.
- Контекст, добавочный номер и метка приоритета, например incoming,123,weasels.

Любое из мест назначения может быть опущено, но не оба одновременно. Если согласно вычислениям переход должен осуществляться по месту назначения, которое не задано, Asterisk просто переходит к следующему приоритету текущего добавочного номера.

Применим GotoIf() в примере:

```
exten => 345,1,Set(TEST=1)
exten => 345,n,GotoIf($[${TEST} = 1]?weasels:iguanas)
exten => 345,n(weasels),Playback(weasels-eaten-phonesys)
exten => 345,n,Hangup()
exten => 345,n(iguanas),Playback(office-iguanas)
exten => 345,n,Hangup()
```

Вы заметите, что за каждым приложением Playback() следует приложение Hangup(). Это делается для того, чтобы при переходе на метку weasels вызов заканчивался до того, как начинается воспроизведение звукового файла office-iguanas. Все чаще можно увидеть добавочные номера, разбитые на несколько компонентов (разделенных между собой командой Hangup()), каждый из которых представляет собой этапы, выполняемые следом за GotoIf().

Обычно при такой схеме, когда требуется оградить Asterisk от выполнения следующего приоритета после выполнения перехода, вероятно, лучше в качестве места назначения использовать другие добавочные номера, а не метки приоритетов. Во всяком случае, это делает диалплан несколько понятнее. Предыдущий фрагмент диалплана можно было бы переписать следующим образом:

```
exten => 345,1,Set(TEST=1)
exten => 345,n,GotoIf($[${TEST} = 1]?weasels,1:iguanas,1); теперь переход
; выполняется к добавочному номеру,приоритету
exten => weasels,1,Playback(weasels-eaten-phonesys); это НЕ метка.
```

### Предоставление условного перехода только на случай ложности выражения

Предыдущий пример можно было бы записать следующим образом:

```
exten => 345,1,Set(TEST=1)
exten => 345,n,GotoIf($[${TEST} = 1]?:iguanas) ; здесь больше нет
; метки weasels, но это будет по-прежнему работать
exten => 345,n,Playback(weasels-eaten-phonesys)
exten => 345,n,Hangup()
exten => 345,n(iguanas),Playback(office-iguanas)
exten => 345,n,Hangup()
```

Между символами ? и : ничего не указано, таким образом, если выражение оказывается истинным, выполнение диалплана перейдет на следующую строку. Поскольку это именно то, что требуется, указывать метку необязательно.

На самом деле мы не рекомендуем этого делать, потому что такая запись сложна для понимания. Но подобные диалпланы встречаются, поэтому надо просто знать, что данный синтаксис тоже является абсолютно правильным.

```
; Это другой добавочный номер
exten => weasels,n,Hangup()
exten => iguanas,1,Playback(office-iguanas)
exten => iguanas,n,Hangup()
```

При изменении значения, присваиваемого TEST в первой строке, сервер Asterisk должен воспроизводить разные приветствия.

Рассмотрим другой пример выполнения переходов по условию. На этот раз будем использовать оба приложения, и Goto(), и GotoIf(). Выполним счет в обратном направлении от 10 и повесим трубку:

```
exten => 123,1,Set(COUNT=10)
exten => 123,n(start),GotoIf($[${COUNT} > 0]?:goodbye)
exten => 123,n,SayNumber(${COUNT})
exten => 123,n,Set(COUNT=$[${COUNT} - 1])
exten => 123,n,Goto(start)
exten => 123,n(goodbye),Hangup()
```

Проанализируем этот пример. В первом приоритете переменной COUNT присваивается значение 10. Далее COUNT сравнивается с 0. Если ее значение больше нуля, мы переходим к следующему приоритету. (Не забываем, что, если место назначения в приложении GotoIf() опущено, управление передается вниз следующему приоритету.) С этого момента воспроизводится номер, из COUNT вычитается 1 и управление опять возвращается к приоритету start (начало). Если значение COUNT меньше или равно 0, управление передается приоритету goodbye (до свидания) и вызов завершается.

Классический пример выполнения переходов по условию с нежностью называют «логикой для защиты от любимой девушки». Если номер Caller ID (ID звонящего) входящего вызова совпадает с номером телефона бывшей девушки абонента, Asterisk выдает сообщение, отличающееся от того, которым он обычно встречает звонки от других абонентов. Несмотря на простоту и примитивность, это хороший пример для изучения переходов по условию в диалплане Asterisk.

В этом примере используется функция `CALLERID`, которая позволяет извлекать информацию о Caller ID (ID звонящего) входящего вызова. Пусть для данного случая номер телефона жертвы будет 888-555-1212:

```
exten => 123,1,GotoIf($[${CALLERID(num)} = 8885551212]?reject:allow)
exten => 123,n(allow),Dial(Zap/4)
exten => 123,n,Hangup()
exten => 123,n(reject),Playback(abandon-all-hope)
exten => 123,n,Hangup()
```

В приоритете 1 вызывается приложение `GotoIf()`. Оно указывает Asterisk перейти к метке приоритета `reject` (отклонить), если номер Caller ID (ID звонящего) соответствует `8885551212`, а в противном случае перейти к метке приоритета `allow` (можно было бы опустить имя метки, позволяя `GotoIf()` просто передать управление далее). Если номер Caller ID (ID звонящего) совпадает с указанным, управление вызовом передается приоритету `reject`, который воспроизводит нежелательному абоненту неприветливое сообщение. В противном случае делается попытка вызова получателя по каналу `Zap/4`.

## Переходы по условию с временным критерием с помощью GotoIfTime()

Другой вариант использования переходов по условию в диалплане – приложение `GotoIfTime()`. `GotoIf()` для принятия решения производит вычисление выражения, тогда как `GotoIfTime()` выбирает, в какую ветвь диалплана выполнить переход, на основании текущего системного времени.

Самое очевидное применение этого приложения – предоставление абонентам разных приветствий до начала рабочего времени и после его окончания.

Приложение `GotoIfTime()` имеет следующий синтаксис:

```
GotoIfTime(times,days_of_week,days_of_month,months?label)
```

Одним словом, `GotoIfTime()` передает вызов в заданную метку *`label`*, если текущие дата и время соответствуют критерию, заданному параметрами *`times`* (время), *`days_of_week`* (дни недели), *`days_of_month`* (дни месяца) и *`months`* (месяцы). Рассмотрим каждый аргумент более подробно:

*`times`*

Это список из одного или более временных диапазонов, записанных в 24-часовом формате. Например, период с 9:00 утра до 5:00 вечера

будет задан так: `09:00-17:00`. День начинается с 0:00 и заканчивается в 23:59.

Следует отметить, что *times* прекрасно выполняет циклические переходы. Поэтому, если требуется задать период, когда офис закрыт, в параметре *times* можно указать `18:00-9:00` – и все будет работать, как ожидается. Обратите внимание, что данная техника применима и для других компонентов `GotoIfTime()`. Например, описать выходные дни можно, задав `sat-sun` (суббота-воскресенье).

*days_of_week*

Это список из одного или более дней недели. Дни должны быть заданы как `mon`, `tue`, `wed`, `thu`, `fri`, `sat` и/или `sun`. Период с понедельника по пятницу можно задать как `mon-fri`. Вторник и четверг были бы представлены как `tue&thu`.

Обратите внимание, что можно задавать сочетание диапазонов и отдельных дней, например `sun-mon&wed&fri-sat` или проще, `wed&fri-mon`.

*days_of_month*

Это список дней месяца. Дни задаются числами от 1 до 31. Период с 7-го до 12-го числа будет представлен как `7-12`, а 15-е и 30-е числа месяца будут записаны как `15&30`.

*months*

Это список из одного или более месяцев года. Диапазон месяцев должен быть записан как `jan-apr`, а если требуется указать месяцы, идущие не по порядку, они записываются через амперсанд, например `jan&mar&jun`. Также можно комбинировать диапазоны и отдельные месяцы: `jan-apr&jun&oct-dec`.

Если необходимо выбрать все возможные значения для любого из этих аргументов, просто задайте `*` для него.

В качестве аргумента *label* может быть задано любое из следующих значений:

- Метка приоритета в рамках того же добавочного номера, например `time_has_passed` (время прошло).
- Добавочный номер и приоритет в рамках того же контекста, например `123,time_has_passed`.
- Контекст, добавочный номер и приоритет, например `incoming,123,time_has_passed`.

Теперь, после того как мы ознакомились с синтаксисом, рассмотрим несколько примеров. Следующий пример соответствует промежутку времени с 9:00 утра до 5:59 вечера, с понедельника по пятницу, любого дня месяца, в любой месяц года:

```
exten => s,1,GotoIfTime(09:00-17:59,mon-fri,*,*?open,s,1)
```

Если абонент звонит в данные часы, вызов будет направлен в первый приоритет добавочного номера `s` контекста `open`. Если вызов поступает в другое время, вне заданного промежутка, он будет направлен в следующий приоритет текущего добавочного номера. Это позволяет выполнять переходы по условию с несколькими временными критериями, как показано в примере ниже (обратите внимание, что более конкретизированные временные диапазоны всегда должны быть указаны перед менее конкретными):

```
; Если это любой час любого дня недели, который
; является четвертым днем месяца июля, мы закрыты
exten => s,1,GotoIfTime(*,*,4,jul?open,s,1)

; В рабочие часы вызовы направляются в контекст open
exten => s,n,GotoIfTime(09:00-17:59|mon-fri|*|*?open,s,1)
exten => s,n,GotoIfTime(09:00-11:59|sat|*|*?open,s,1)

; В противном случае мы закрыты
exten => s,n,Goto(closed,s,1)
```

Возможна ситуация, когда для интервала задано конечное время 17:58, а текущее время – 17:59, но диалплан не меняет своего поведения. Для такого случая следует заметить, что степень дискретизации приложения `GotoIfTime()` равна двум минутам. Поэтому, если в качестве времени завершения периода задано 18:00, система изменит свое поведение только в 18:01:59.

## Голосовая почта

Одной из самых популярных (или, возможно, непопулярных) функций любой современной системы телефонной связи является голосовая почта. Естественно, Asterisk обладает достаточно гибкой системой голосовой почты. Среди ее возможностей можно упомянуть:

- Неограниченные по размеру защищенные паролем ящики голосовой почты, в каждом из которых содержатся почтовые папки для организации сообщений голосовой почты.
- Различные приветствия для состояний «занято» и «недоступен».
- Стандартные и специальные приветствия.
- Возможность связывать телефоны с несколькими почтовыми ящиками, а почтовые ящики – с несколькими телефонами.
- Уведомление о поступлении сообщения голосовой почты по электронной почте с возможностью прикрепления сообщения голосовой почты в виде звукового файла[1].
- Пересылка и широковещательные рассылки голосовой почты.

[1] Нет, за это действительно не надо платить, и да, это действительно работает.

- Индикатор ожидающих сообщений (мигающий световой индикатор или прерывистый тональный сигнал) во многих типах телефонов.
- Каталог ящиков голосовой почты служащих компании.

И это только вершина айсберга! В данном разделе будут представлены основы типовой настройки голосовой почты.

Конфигурация голосовой почты описывается в конфигурационном файле voicemail.conf. Этот файл содержит набор настроек, которые могут использоваться для приведения системы голосовой почты в соответствие конкретным требованиям. Рассмотреть все доступные опции voicemail.conf в одной главе было бы просто невозможно, но образец конфигурационного файла хорошо задокументирован и прост для понимания. Пока что давайте посмотрим в конец файла, где описываются контексты и ящики голосовой почты.

Контексты диалплана просто разделяют разные части диалплана, контексты голосовой почты позволяют определять разные, изолированные друг от друга, наборы почтовых ящиков. Это обеспечивает возможность хранить голосовую почту нескольких разных компаний или офисов на одном сервере. Контексты голосовой почты определяются так же, как и контексты диалплана: с помощью имени контекста, заключенного в квадратные скобки. В примерах данной книги будет использоваться контекст голосовой почты `[default]`.

## Создание почтовых ящиков

Внутри каждого контекста голосовой почты определяются разные почтовые ящики. Для описания почтового ящика используется следующий синтаксис:

*почтовыйящик* => *пароль*,*имя*[,*email*[,*email_пейджера*[,*опции*]]]

Поясним назначение каждой части описания почтового ящика:

*почтовыйящик*

Это номер почтового ящика. Обычно он соответствует добавочному номеру ассоциированного с ним телефонного аппарата.

*пароль*

Числовой пароль, который будет использоваться владельцем почтового ящика для доступа к своей голосовой почте. Если пользователь изменит свой пароль, система обновит это поле в файле voicemail.conf.

*имя*

Это имя владельца почтового ящика. Текст данного поля используется в телефонном справочнике компании для обеспечения возможности абонентам при вызове применять имена пользователей.

*email*

Это адрес электронной почты владельца ящика голосовой почты. Asterisk может посылать уведомления о получении голосовой почты

(включая само сообщение голосовой почты) на заданный ящик электронной почты.

*email_пейджера*

Это электронный адрес пейджера или мобильного телефона владельца ящика голосовой почты. Asterisk может посылать короткое сообщение, уведомляющее о получении голосовой почты, на заданный адрес электронной почты.

*опции*

Это список опций, определяющих часовой пояс, в котором находится владелец почтового ящика, и переопределяющих глобальные настройки голосовой почты. Существует девять допустимых опций: `attach`, `serveremail`, `tz`, `saycid`, `review`, `operator`, `callback`, `dialout` и `exitcontext`. Эти опции определяются парами *опция=значение*, разделяемыми символами вертикальной черты (`|`). Опция `tz` задает часовой пояс пользователя соответственно часовому поясу, определенному ранее в разделе `[zonemessages]` файла voicemail.conf. Остальные восемь опций переопределяют соответствующие их именам глобальные настройки голосовой почты.

Вот типовое описание почтового ящика:

```
101 => 1234,Joe Public,jpublic@somedomain.com,jpublic@pagergateway.net,
tz=central|attach=yes
```

Продолжая создание нашего диалплана из последней главы, зададим ящики голосовой почты для Джона и Джейн. Для Джона определим пароль 1234, а для Джейн – 4444 (помните, это делается в файле voicemail.conf, а не в extensions.conf):

```
[default]
101 => 1234,John Doe,john@asteriskdocs.org,jdoe@pagergateway.tld
102 => 4444,Jane Doe,jane@asteriskdocs.org,jane@pagergateway.tld
```

## Добавление голосовой почты в диалплан

Теперь, когда созданы почтовые ящики для Джейн и Джона, давайте обеспечим возможность абонентам оставлять сообщения для них в случае, если они не отвечают на звонок. Для этого воспользуемся приложением `VoiceMail()`.

Приложение `VoiceMail()` направляет вызывающего абонента на заданный почтовый ящик, чтобы он мог оставить сообщение. Почтовый ящик должен быть описан так: *почтовыйящик*@*контекст*, где *контекст* – имя контекста голосовой почты. Сюда также могут быть добавлены буквы `b` или `u` для запроса типа приветствия. Если используется буква `b`, абонент услышит сообщение о *занятости* владельца почтового ящика. Если используется буква `u`, абонент услышит сообщение о *недоступности* владельца почтового ящика (если таковое существует).

Применим это в нашем диалплане. Ранее в контексте `[internal]` располагалась следующая строка, обеспечивающая возможность звонить Джону:

```
exten => 101,1,Dial(${JOHN})
```

Теперь давайте добавим сообщение о недоступности, которое услышит вызывающий абонент, если Джон не ответит на звонок в течение 10 с. Помните, второй аргумент в приложении `Dial()` – время ожидания. Если до истечения времени ожидания ответа на вызов не последовало, звонок перенаправляется в следующий приоритет. Зададим время ожидания 10 с и добавим приоритет для перевода абонента на голосовую почту, в случае если Джон не отвечает на звонок вовремя:

```
exten => 101,1,Dial(${JOHN},10)
exten => 101,n,VoiceMail(101@default,u)
```

Теперь скорректируем это так, чтобы, если Джон занят (в другом звонке), абонент перенаправлялся на голосовую почту, где слышал бы сообщение о занятости. Для этого воспользуемся переменной `${DIALSTATUS}`, в которой содержится одно из значений статуса (список всех возможных значений можно получить в консоли Asterisk с помощью команды `core show application dial`):

```
exten => 101,1,Dial(${JOHN},10)
exten => 101,n,GotoIf($["${DIALSTATUS}" = "BUSY"]?busy:unavail)
exten => 101,n(unavail),Voicemail(101@default,u)
exten => 101,n,Hangup()
exten => 101,n(busy),VoiceMail(101@default,b)
exten => 101,n,Hangup()
```

Теперь абоненты будут получать сообщение голосовой почты Джона (с соответствующим приветствием), если Джон занят или недоступен. Однако осталась небольшая проблема – Джон не имеет возможности извлекать свои сообщения. Исправим это.

## Организация доступа к голосовой почте

Пользователи могут извлекать свои сообщения голосовой почты, менять опции и записывать приветствия голосовой почты с помощью приложения `VoiceMailMain()`. В своей типовой форме приложение `VoiceMailMain()` вызывается без аргументов. Добавим в контекст `[internal]` диалплана добавочный номер 700, чтобы внутренние пользователи могли выполнять звонки для доступа к своим сообщениям голосовой почты:

```
exten => 700,1,VoiceMailMain()
```

## Создание телефонного справочника для набора номера по имени

Осталась последняя заслуживающая внимания функция системы голосовой почты Asterisk – телефонный справочник для набора номера по имени. Создается он с помощью приложения `Directory()`. Это приложение, используя имена, заданные в описаниях почтовых ящиков в файле voicemail.conf, предоставляет абоненту телефонный справочник абонентов АТС для набора номеров по имени.

Directory() принимает три аргумента: контекст голосовой почты, из которого считываются имена, контекст диалплана, в котором вызывается пользователь (необязательный), и строку опций (также необязательный). По умолчанию Directory() ведет поиск пользователя по фамилии. Если передается опция f, поиск осуществляется по имени. Добавим два справочника для набора номера по имени в контекст [incoming] нашего образца диалплана, чтобы абоненты могли выполнять поиск или по имени, или по фамилии:

```
exten => 8,1,Directory(default,incoming,f)
exten => 9,1,Directory(default,incoming)
```

Если абоненты нажмут кнопку 8, они получат справочник, составленный по именам. Если они наберут 9, то получат справочник, составленный по фамилиям.

## Макрос

Макросы[1] – очень полезная структура, разработанная во избежание повторов в диалплане. Также они облегчают задачу по внесению изменений в диалплан. Чтобы проиллюстрировать это, снова вернемся к нашему примеру диалплана. Если вы помните, при внесении изменений для голосовой почты мы остановились на следующем варианте добавочного номера для Джона:

```
exten => 101,1,Dial(${JOHN},10)
exten => 101,n,GotoIf($["${DIALSTATUS}" = "BUSY"]?busy:unavail)
exten => 101,n(unavail),Voicemail(101@default,u)
exten => 101,n,Hangup()
exten => 101,n(busy),VoiceMail(101@default,b)
exten => 101,n,Hangup()
```

Допустим, системой Asterisk пользуются сто абонентов. Для задания добавочных номеров пришлось бы многократно копировать и вставлять фрагменты кода. Теперь предположим, что возникла необходи-

[1] Макрос похож на подпрограмму диалплана общего назначения, но ему свойственна проблема переполнения стека, поэтому не следует создавать в макросах вызовы с более чем пятикратной глубиной вложенности. На момент написания данной книги мы не располагаем информацией о том, будет ли приложение Macro доработано для версии 1.4 или переписано для будущих версий. Если планируется выполнять множество вложенных макросов (и вызывать в их рамках сложные функции), есть вероятность возникновения нестабильности. Проблему выявит всего лишь один тестовый звонок, поэтому, если диалплан проходит тестирование, он готов к работе. Также рекомендуется обратить внимание на приложения Gosub и Return, поскольку зачастую функциональность, реализуемая с помощью Macro(), на самом деле может быть получена без его использования. Также, будьте добры, отметьте, что мы не предлагаем не использовать Macro(). Это фантастический инструмент, обладающий замечательными рабочими характеристиками; ему просто свойственны некоторые проблемы с вложенностью.

мость немного скорректировать работу добавочных номеров. Для этого пришлось бы вносить массу изменений, и почти наверняка при этом будут допущены ошибки.

Вместо этого можно определить макрос, содержащий список необходимых шагов, и затем сделать так, чтобы все добавочные номера использовали его. В этом случае корректировать придется только макрос, и это обеспечит изменение всех элементов диалплана, ссылающихся на него.

Те, кто хорошо знаком с программированием, заметят, что макросы подобны подпрограммам, которые есть во многих современных языках программирования. Если вы не знакомы с программированием, не волнуйтесь, мы подробно и последовательно ознакомим вас с созданием макроса.

Лучший способ оценить макрос – увидеть его в деле, так что прямо сейчас и приступим.

## Описание макроса

Давайте возьмем логику диалплана, которая использовалась выше для настройки голосовой почты Джона, и превратим ее в макрос. Затем с помощью этого макроса обеспечим Джону и Джейн (и их коллегам) аналогичные функциональные возможности.

Описание макроса очень похоже на контексты. (По сути, можно даже утверждать, что это действительно небольшие контексты с ограниченной функциональностью.) Для описания макроса используется служебное слово `macro-`, за которым следует имя макроса, и вся эта конструкция заключается в квадратные скобки:

```
[macro-voicemail]
```

Имена макросов должны начинаться со служебного слова `macro-`. Это отличает их от обычных контекстов. Команды внутри макроса формируются практически аналогично всем остальным элементам диалплана; единственное ограничение – макросы используют только добавочный номер `s`. Введем логику голосовой почты в макрос, заменяя по ходу добавочные номера на `s`:

```
[macro-voicemail]
exten => s,1,Dial(${JOHN},10)
exten => s,n,GotoIf($["${DIALSTATUS}" = "BUSY"]?busy:unavail)
exten => s,n(unavail),Voicemail(101@default,u)
exten => s,n,Hangup()
exten => s,n(busy),VoiceMail(101@default,b)
exten => s,n,Hangup()
```

Для начала хорошо, но не идеально, потому что данный макрос предназначен только для Джона и номера его почтового ящика. Чтобы сделать его универсальным и подходящим не только для Джона, но и для всех его сослуживцев, воспользуемся другой возможностью макроса – его аргументами. Но прежде рассмотрим, как осуществлять вызов макроса в диалплане.

## Вызов макроса из диалплана

Использовать макрос в диалплане нам поможет приложение `Macro()`. Оно вызывает заданный макрос и передает в него необходимые аргументы. Например, чтобы вызвать наш макрос голосовой почты из диалплана, можно сделать следующее:

```
exten => 101,1,Macro(voicemail)
```

Приложение `Macro()` определяет также несколько специальных переменных. К ним относятся:

`${MACRO_CONTEXT}`

Исходный контекст, в котором был вызван макрос.

`${MACRO_EXTEN}`

Исходный добавочный номер, в котором был вызван макрос.

`${MACRO_PRIORITY}`

Исходный приоритет, в котором был вызван макрос.

`${ARG` *n*`}`

*n*-ный аргумент, передаваемый в макрос. Например, первым был бы аргумент `${ARG1}`, вторым – `${ARG2}` и т. д.

Как говорилось ранее, описанный выше макрос был жестко определен для Джона, тогда как должен быть универсальным. Давайте изменим его и вместо номера почтового ящика 101 будем использовать переменную `${MACRO_EXTEN}`. Таким образом, при вызове макроса из добавочного номера 101 сообщения голосовой почты будут приходить в почтовый ящик 101; если макрос будет вызван из добавочного номера 102, сообщения пойдут в почтовый ящик 102, и т. д.:

```
[macro-voicemail]
exten => s,1,Dial(${JOHN},10)
exten => s,n,GotoIf($["${DIALSTATUS}" = "BUSY"]?busy:unavail)
exten => s,n(unavail),Voicemail(${MACRO_EXTEN}@default,u)
exten => s,n,Hangup()
exten => s,n(busy),VoiceMail(${MACRO_EXTEN}@default,b)
exten => s,n,Hangup()
```

## Использование аргументов в макросе

Итак, мы уже близки к получению того макроса, какой нам нужен, осталось внести последнее изменение. Необходимо передать в макрос канал, по которому будут выполняться вызовы, потому что до сих пор в нем жестко прописан канал `${JOHN}` (помните, мы определяли переменную `JOHN` как канал для звонков Джону?). Передадим канал как аргумент – и наш первый макрос будет готов:

```
[macro-voicemail]
exten => s,1,Dial(${ARG1},10)
exten => s,n,GotoIf($["${DIALSTATUS}" = "BUSY"]?busy:unavail)
exten => s,n(unavail),Voicemail(${MCARO_EXTEN}@default,u)
```

```
exten => s,n,Hangup()
exten => s,n(busy),VoiceMail(${MCARO_EXTEN}@default,b)
exten => s,n,Hangup()
```

Теперь, когда макрос готов, его можно использовать в диалплане. Вот как вызывается макрос для обеспечения функции голосовой почты для Джона, Джейн и Джека:

```
exten => 101,1,Macro(voicemail,${JOHN})
exten => 102,1,Macro(voicemail,${JANE})
exten => 103,1,Macro(voicemail,${JACK})
```

И для 50 или более пользователей этот диалплан сохранит свою четкость и организованность; для каждого пользователя просто будет создана строка со ссылкой на макрос, который может быть настолько сложным, насколько это необходимо. Может существовать даже несколько разных макросов для различных типов пользователей, таких как `executives` (руководство), `courtesy_phones` (телефоны справочной службы), `call_center_agents` (агенты центра обработки вызовов), `analog_sets` (аналоговые телефоны), `sales_department` (отдел продаж) и т. д.

Более сложная версия макроса может выглядеть примерно так:

```
[macro-voicemail]
exten => s,1,Dial(${ARG1},20)
exten => s,n,Goto(s-${DIALSTATUS},1)
exten => s-NOANSWER,1,Voicemail(${MACRO_EXTEN},u)
exten => s-NOANSWER,n,Goto(incoming,s,1)
exten => s-BUSY,1,Voicemail(${MACRO_EXTEN},b)
exten => s-BUSY,n,Goto(incoming,s,1)
exten => _s-.,1,Goto(s-NOANSWER,1)
```

Этот макрос использует замечательный побочный эффект приложения `Dial()`: `Dial()` задает переменную `DIALSTATUS` как индикатор успешности или неуспешности вызова. В данном случае обрабатываются варианты `NOANSWER` (не отвечает) и `BUSY` (занято), а все остальные результаты трактуются как `NOANSWER`.

# Использование базы данных Asterisk (AstDB)

Ну что, пока довольны? Дальше будет еще интереснее!

В Asterisk есть мощный механизм для хранения значений, который называется базой данных Asterisk (AstDB). AstDB обеспечивает простой способ хранения данных для использования в диалплане.

Для тех, кто имеет опыт использования реляционных баз данных, таких как PostgreSQL или MySQL, заметим, что база данных Asterisk не является традиционной реляционной базой данных. Это база данных Berkeley DB версии 1. Существует несколько способов хранения данных из Asterisk в реляционной базе данных. Подробнее о реляционных базах данных рассказывается в главе 12.

База данных Asterisk хранит данные в группах, называемых *семействами*, значения которых идентифицируются *ключами*. Например, если бы у нас имелось семейство `test`, мы могли бы хранить только одно значение с ключом `count`. Каждое хранящееся значение должно быть ассоциировано с каким-то семейством.

## Хранение данных в AstDB

Для сохранения нового значения в базе данных Asterisk используется приложение `Set()`[1], но с его помощью задается переменная не канала, а AstDB. Например, чтобы присвоить ключу `count` семейства `test` значение 1, необходимо записать следующее:

```
exten => 456,1,Set(DB(test/count)=1)
```

Если в семействе `test` уже есть ключ `count`, его значение будет заменено на новое. Также можно сохранять значения из командной строки Asterisk, выполнив команду `database put` *семейство ключ значение*. Для нашего примера надо ввести следующее: `data base put test count 1`.

## Извлечение данных из AstDB

Чтобы извлечь значение из базы данных Asterisk и присвоить его переменной, используется все то же приложение `Set()`. Давайте извлечем значение `count` (опять же из семейства `test`), присвоим его переменной `COUNT` и затем воспроизведем это значение абоненту:

```
exten => 456,1,Set(DB(test/count)=1)
exten => 456,n,Set(COUNT=${DB(test/count)})
exten => 456,n,SayNumber(${COUNT})
```

Также можно проверять значение заданного ключа из командной строки Asterisk, выполнив команду `database get` *семейство ключ*. Для просмотра всего содержимого AstDB используется команда `database show`.

## Удаление данных из AstDB

Существует два способа удаления данных из базы данных Asterisk. Чтобы удалить ключ, можно использовать приложение `DB_DELETE()`. Оно принимает путь к ключу в качестве аргумента:

```
; удаляет ключ и возвращает его значение за один шаг
exten => 457,1,Verbose(0, The value was ${DB_DELETE(test/count)})
```

Также можно удалить все семейство ключей, используя приложение `DBdeltree()`. Оно принимает всего один аргумент – имя семейства ключей, подлежащего удалению. Чтобы удалить все семейство `test`, делаем следующее:

```
exten => 457,1,DBdeltree(test)
```

[1] В предыдущих версиях Asterisk были приложения `DBput()` и `DBget()`, которые использовались для задания и извлечения значения из AstDB. Если вы используете старую версию Asterisk, применяйте эти приложения.

Для удаления ключей и семейств ключей AstDB из командной строки используются команды `database del` *ключ* и `database deltree` *семейство* соответственно.

## Использование AstDB в диалплане

Существует бесконечное число вариантов использования базы данных Asterisk в диалплане. Чтобы представить AstDB, рассмотрим два простых примера. Первый – простой пример вычисления, демонстрирующий постоянство базы данных Asterisk (имеется в виду, что она устойчива к перезагрузкам системы). Во втором примере с помощью функции `BLACKLIST()` будет определена принадлежность номера к черному списку и необходимость его блокировки.

Для начала в примере с воспроизведением счета извлечем число (значение ключа `count`) из базы данных и присвоим его переменной `COUNT`. Если такой ключ не существует, `DB()` возвратит `NULL` (нет значения). Чтобы проверить наличие значения в базе данных, введем функцию `ISNULL()`, которая будет контролировать, возвращено ли значение. В случае если этого не произошло, мы присвоим AstDB исходное значение 1 с помощью приложения `Set()`. Следующий приоритет возвратит нас к приоритету 1. Это произойдет при первом вызове данного добавочного номера:

```
exten => 678,1,Set(COUNT=${DB(test/count)})
exten => 678,n,GotoIf($[${ISNULL(${COUNT})}]?:continue)
exten => 678,n,Set(DB(test/count)=1)
exten => 678,n,Goto(1)
exten => 678,n(continue),NoOp()
```

Далее будет воспроизведено текущее значение `COUNT`. После этого оно будет увеличено на 1:

```
exten => 678,1,Set(COUNT=${DB(test/count)})
exten => 678,n,GotoIf($[${ISNULL(${COUNT})}]?:continue)
exten => 678,n,Set(DB(test/count)=1)
exten => 678,n,Goto(1)
exten => 678,n(continue),NoOp()
exten => 678,n,SayNumber(${COUNT})
exten => 678,n,Set(COUNT=$[${COUNT} + 1])
```

Теперь, после приращения `COUNT`, давайте поместим новое значение в базу данных. Не забудьте, что сохранение значения для существующего ключа приводит к перезаписи предыдущего значения:

```
exten => 678,1,Set(COUNT=${DB(test/count)})
exten => 678,n,GotoIf($[${ISNULL(${COUNT})}]?:continue)
exten => 678,n,Set(DB(test/count)=1)
exten => 678,n,Goto(1)
exten => 678,n(continue),NoOp()
exten => 678,n,SayNumber(${COUNT})
exten => 678,n,Set(COUNT=$[${COUNT} + 1])
exten => 678,n,Set(DB(test/count)=${COUNT})
```

Наконец вернемся к первому приоритету. Теперь приложение будет продолжать счет:

```
exten => 678,1,Set(COUNT=${DB(test/count)})
exten => 678,n,GotoIf($[${ISNULL(${COUNT})}]?:continue)
exten => 678,n,Set(DB(test/count)=1)
exten => 678,n,Goto(1)
exten => 678,n(continue),NoOp()
exten => 678,n,SayNumber(${COUNT})
exten => 678,n,Set(COUNT=$[${COUNT} + 1]
exten => 678,n,Set(DB(test/count)=${COUNT})
exten => 678,n,Goto(1)
```

Попробуйте этот пример. Послушайте немного, как приложение считает, и повесьте трубку. Когда вы наберете этот добавочный номер снова, счет должен быть продолжен с той цифры, на которой вы остановились. Значение, хранящееся в базе данных, будет постоянным даже при перезагрузке Asterisk.

В следующем примере логика диалплана будет организована вокруг функции `BLACKLIST()`, которая проверяет наличие Caller ID (ID звонящего) текущего абонента в черном списке. (Черный список – это просто семейство AstDB, называемое `blacklist`.) Если функция `BLACKLIST()` находит номер в черном списке, она возвращает значение 1, в противном случае возвращается 0. Эти значения в сочетании с приложением `GotoIf()` могут использоваться для управления выполнением приложения `Dial()` при вызове:

```
exten => 124,1,GotoIf($[${BLACKLIST()]?blocked,1)
exten => 124,n,Dial(${JOHN})
exten => blocked,1,Playback(privacy-you-are-blacklisted)
exten => blocked,n,Playback(vm-goodbye)
exten => blocked,n,Hangup()
```

Чтобы добавить номер в черный список, выполните команду `database put blacklist` *номер* `1` из интерфейса командной строки Asterisk.

# Полезные функции Asterisk

Теперь, рассмотрев дополнительные базовые возможности, давайте перейдем к ряду популярных функций, включенных в Asterisk.

## Zapateller()

`Zapateller()` – это простое приложение Asterisk, которое воспроизводит специальный информационный тон в начале звонка. Устройства автоматического набора (обычно используемые в системах продаж по телефону) принимают этот тон за сигнал разъединения линии. Причем они не только прекратят вызов, но также пометят данный номер как не обслуживаемый, что поможет избежать всех видов телемаркетинговых звонков. Чтобы использовать эту функциональность в своем диалплане, вам надо просто вызвать приложение `Zapateller()`.

Также применим необязательную опцию `nocallerid`, чтобы тон воспроизводился только в случае, если входящий вызов не предоставляет информации о Caller ID (ID звонящего). Вот пример использования приложения `Zapateller()` в добавочном номере контекста `[incoming]`:

```
[incomimg]
exten => s,1,Zapateller(nocallerid)
exten => s,n,Playback(enter-ext-of-person)
```

## Парковка вызова

Еще одна удобная функция – парковка вызова. Она обеспечивает возможность перевести вызов в состояние ожидания, поставить его на «парковку», чтобы он мог быть принят на другом добавочном номере. Все параметры парковки вызовов (такие, как используемые добавочные номера, количество мест и т. д.) задаются в конфигурационном файле features.conf. Раздел `[general]` файла features.conf содержит четыре настройки, касающиеся парковки вызовов:

`parkext`

Это добавочный номер для парковки. Передайте вызов на этот добавочный номер – и система сообщит, в какой парковочный слот он помещен. Добавочный номер для парковки по умолчанию – 700.

`parkpos`

Эта опция определяет количество парковочных слотов. Например, задав номера 701–720, вы создадите 20 парковочных слотов с нумерацией от 701 до 720.

`context`

Это имя контекста парковки. Чтобы иметь возможность парковать вызовы, необходимо включить этот контекст.

`parkingtime`

Если эта опция задана, она определяет, как долго (в секундах) вызов может оставаться на парковке. Если вызов не принят в течение заданного времени, выполняется звонок на добавочный номер, с которого вызов поступил на парковку.

После редактирования файла features.conf необходимо перезагрузить Asterisk, потому что чтение этого файла выполняется только при запуске системы. Выполнение команды `reload` не обеспечит чтения файла features.conf.

Также обратите внимание, что, поскольку пользователю необходимо иметь возможность переводить вызовы на добавочный номер парковки, в приложении `Dial()` должны использоваться опции `t` и/или `T`.

Итак, давайте создадим простой диалплан для демонстрации парковки вызовов:

```
[incoming]
include => parkedcalls
```

```
exten => 103,1,Dial(SIP/Bob,,tT)
exten => 104,1,Dial(SIP/Charlie,,tT)
```

Проиллюстрируем принцип работы парковки вызовов. Скажем, Элис звонит в систему и набирает добавочный номер 103, чтобы поговорить с Бобом. Через некоторое время Боб переводит вызов на добавочный номер 700, который сообщает ему, что звонок от Элис был припаркован в слот 701. После этого Боб звонит Чарли на добавочный номер 104 и говорит ему, что Элис ожидает по номеру 701. Чарли набирает добавочный номер 701 и разговаривает с Элис. Это простой и эффективный способ обеспечить возможность переключения вызывающих абонентов между пользователями системы.

Аргументы `t` и `T` приложения `Dial()` нужны не для всех типов каналов. Например, многие SIP-телефоны реализуют это с помощью функциональной или обычной кнопки и обмена сигналами по протоколу SIP.

## Организация конференц-связи с помощью MeetMe()

Не менее полезной функцией является установление аудиоконференц-связи с помощью приложения `MeetMe()`[1]. Это приложение обеспечивает возможность одновременного общения множества абонентов так, как если бы они все физически находились в одном месте. К основным функциям относятся:

- Возможность создания защищенных паролем конференций.
- Администрирование конференции (отключение звука конференции, блокировка конференции, исключение участников).
- Опция отключения звука всех участников, кроме одного (полезна для объявлений по компании, широковещательных рассылок и т. д.).
- Создание статических или динамических конференций.

Давайте поэтапно рассмотрим процесс настройки базового конференц-зала. Конфигурационные опции для системы конференц-связи `MeetMe` располагаются в файле meetme.conf. В этом конфигурационном файле задаются конференц-залы и необязательные числовые пароли. (Если пароль задан, он будет необходим для входа на все конференции, проводимые с использованием этого конференц-зала.) Для нашего примера настроим конференц-зал по добавочному номеру 600. Сначала зададим все настройки в файле meetme.conf. Назовем этот конференц-зал `600` и на этот раз не будем задавать пароль:

```
[rooms]
conf => 600
```

[1] В мире традиционных офисных АТС этот тип функциональности очень дорог. Приходится или выкладывать бешеные суммы за внешний сервис обеспечения конференц-связи, или доукомплектовывать специализированную офисную АТС дорогим коммутатором для такой возможности.

Закончив работу с конфигурационным файлом, необходимо перезагрузить Asterisk, чтобы она могла повторно прочитать файл meetme.conf. Далее добавим поддержку конференц-зала в диалплан, используя приложение MeetMe(). MeetMe() принимает три аргумента: имя конференц-зала (заданное в meetme.conf), набор опций и пароль, который пользователь должен ввести, чтобы присоединиться к конференции. Настроим простую конференцию, используя конференц-зал 600, опцию i (которая обеспечивает оповещение о том, что кто-то присоединился или покинул конференцию) и пароль 54321:

```
exten => 600,1,MeetMe(600,i,54321)
```

Вот и все! Когда абоненты попадут на добавочный номер 600, им будет предложено ввести пароль. Если они правильно введут 54321, то попадут на конференцию. Полный список всех опций, поддерживаемых приложением MeetMe(), представлен в приложении B.

Другое полезное приложение – MeetMeCount(). Как следует из его имени, это приложение подсчитывает, сколько пользователей находится в том или ином конференц-зале. Оно принимает два аргумента: конференц-зал, где необходимо подсчитать количество участников, и необязательное имя переменной, в которой нужно сохранить это число. Если второй аргумент, то есть имя переменной, не задан, полученное число воспроизводится вызывающему абоненту:

```
exten => 601,1,Playback(conf-thereare)
exten => 601,n,MeetMeCount(600)
exten => 601,n,Playback(conf-peopleinconf)
```

Если вторым аргументом в MeetMeCount() передается переменная, итоговое количество участников присваивается этой переменной, а само число не воспроизводится. Так можно ограничивать количество участников:

```
; ограничить конференц-зал 10 участниками
exten => 600,1,MeetMeCount(600,CONFCOUNT)
exten => 600,n,GotoIf($[${CONFCOUNT} <= 10]?meetme:conf_full,1)
exten => 600,n(meetme),MeetMe(600,i,54321)

exten => conf_full,1,Playback(conf-full)
```

Разве Asterisk не забавна?

# Заключение

В данной главе было рассмотрено еще несколько приложений диалплана Asterisk. Надеемся, что это обеспечит вам фундамент, на основе которого можно экспериментировать с созданием собственных диалпланов. Как и в предыдущей главе, мы рекомендуем вернуться назад и перечитать разделы, в которых для вас остались неясные моменты.

В следующих главах мы немного отвлечемся от Asterisk, чтобы поговорить о некоторых технологиях, используемых во всех системах телефонной связи. Мы будем часто упоминать Asterisk, но то, что мы собираемся обсуждать, в большой мере присуще многим телекоммуникационным системам.

# 7

# Что такое телефония

*Один телефон – это необходимость, два телефона – богатство, три телефона – роскошь, а ни одного телефона – блаженство.*

– Дуг Ларсон

Теперь мы собираемся отдохнуть от Asterisk на протяжении пары главы, потому что хотим посвятить некоторое время обсуждению технологий, с которыми придется взаимодействовать системе Asterisk. В данной главе мы поговорим о некоторых технологиях традиционной телефонной сети, особенно о тех, которые люди чаще всего хотят подключить к Asterisk. (Передача голоса по IP-протоколу обсуждается в следующей главе.)

О технологиях, используемых в телекоммуникационных сетях, можно было бы написать многотомные книги, но материал для данной главы был отобран на основании нашего опыта работы в сообществе, который помог определить круг вопросов, представляющих наибольшую ценность. Хотя эти знания могут и не потребоваться непосредственно для конфигурации системы Asterisk, они будут очень полезны при соединении с системами (и общении с людьми) из мира традиционных телекоммуникаций.

## Аналоговая телефония

Назначение коммутируемой телефонной сети общего пользования (Public Switched Telephone Network, PSTN) – установление и обслуживание аудиосвязи между двумя конечными точками с целью передачи речи.

Хотя люди могут воспринимать звуковые колебания частотой в диапазоне 20–20 000 Гц[1], большинство создаваемых нами звуков располагаются в диапазоне 250–3000 Гц. Поскольку назначение телефонной сети – передача человеческой речи, она была разработана с полосой пропускания примерно 3–3500 Гц. Такой ограниченный частотный диапазон означает, что некоторое качество звука теряется (что может засвидетельствовать любой, кому приходилось слушать музыку, проигрываемую во время ожидания), особенно на высоких частотах.

## Составляющие части аналогового телефона

Аналоговый телефон состоит из пяти частей: звонка, номеронабирателя, гибридного (или сетевого) трансформатора, рычажного переключателя и трубки (оба подключаются к гибридному трансформатору). Звонок, номеронабиратель и гибридный трансформатор могут работать абсолютно независимо друг от друга.

### Звонок

Когда центральная АТС хочет сообщить о входящем звонке, она подает в сеть сигнал переменного тока напряжением примерно 90 В. Это заставит зазвонить колокольчик в вашем телефоне. (В электронных телефонах звонковое устройство может представлять собой не колокольчик, а небольшой электронный частотный модулятор. В конце концов, в качестве звонка может использоваться все, что может реагировать на напряжение звонка; например, в шумных местах, таких как заводы, часто применяются стробирующие световые сигналы.)

Напряжение звонка может быть высоким. При работе с подключенной телефонной линией необходимо обязательно предпринимать меры предосторожности.

Многие путают напряжение переменного тока, активирующее звонок, с напряжением постоянного тока (direct current, DC), питающим телефон. Просто запомните, что для активации колебаний звонка необходим переменный ток (так же как и церковный колокол не будет звонить, если не сообщить ему движение).

В Северной Америке количество звонковых устройств, которые можно подключить к линии, зависит от коэффициента эквивалентности звон-

[1] Если вы хотите увидеть, как выглядят волны с разной частотой на осциллографе, воспользуйтесь программой Sound Frequency Analyzer от компании Reliable Software. Это на самом деле простое и забавное средство, позволяющее «увидеть» звук. Спектрограф дает хорошее изображение сложных гармонических колебаний, генерируемых нашим голосом, а также позволяет различать фоновые звуки, которые всегда окружают нас. Также советуем попробовать раздражающий, но при этом восхитительный NCH Tone Generator от компании NCH Swift Sound.

ка (Ringer Equivalence Number, REN) используемых устройств. Общий REN всех устройств, подключенных к линии, не может превышать 5,0. Старые аналоговые телефоны с электромеханическим звонком имеют REN 1,0. REN некоторых электронных телефонов равен 0,3 или даже меньше. Если подключить слишком много устройств, для питания которых будет необходимо большое напряжение, ни одно из них не сможет генерировать звонок.

## Номеронабиратель

При выполнении телефонного звонка необходимо как-то сообщить сети адрес стороны, с которой вы хотите связаться. Номеронабиратель – это часть телефона, предоставляющая такую функциональность. На заре развития PSTN номеронабиратели были роторными устройствами, которые определяли введенные цифры по импульсам. Это был довольно медленный процесс, поэтому телефонные компании в конце концов перешли на тональный набор. Номеронабиратель для тонального набора, также известного как двухканальный многочастотный (Dual-Tone Multi Frequency, DTMF), состоит из 12 кнопок. За каждой кнопкой закреплено две частоты (табл. 7.1).

*Таблица 7.1. Символы DTMF*

| | **1209 Гц** | **1336 Гц** | **1477 Гц** | **1633 Гц*** |
|---|---|---|---|---|
| 697 Гц | 1 | 2 | 3 | A |
| 770 Гц | 4 | 5 | 6 | B |
| 852 Гц | 7 | 8 | 9 | C |
| 941 Гц | * | 0 | # | D |

* Обратите внимание, что в этом столбце располагаются буквы, которые обычно не представлены на номеронабирателе телефона. Тем не менее они являются частью стандарта DTMF и любой качественный телефон имеет электронику, необходимую для их формирования, даже если на нем нет таких кнопок. (На самом деле такие кнопки можно увидеть на некоторых телефонах, которые преимущественно используются в армии и правительственных структурах.)

При нажатии кнопки номеронабирателя по линии передаются два сигнала соответствующих частот. Дальний конец линии связи может анализировать эти частоты и определять, какой символ был нажат.

## Гибридный (или сетевой) трансформатор

Гибрид – это тип трансформатора, который решает задачу по комбинированию сигналов, передаваемых и получаемых по паре проводов в PSTN и двум парам проводов в трубке. Одна из функций гибрида – регулирование *местного эффекта* (sidetone), то есть того, какая часть передаваемого сигнала возвращается в динамик телефонной трубки. Его назначение – обеспечить связь с более естественным звучанием. Если местного эффекта будет слишком много, абонент будет слышать

свой голос чересчур громко; очень мало местного эффекта – и абонент решит, что линия отключена.

**Рычажный переключатель.** Это устройство сигнализирует о состоянии телефонной цепи на центральную АТС. Когда абонент снимает трубку своего телефона, рычажный переключатель замыкает цепь между ним и центральной АТС, что выглядит как запрос на получение тонального сигнала готовности линии. Когда абонент вешает трубку, рычажный переключатель размыкает цепь, что свидетельствует о завершении вызова[1].

Рычажный переключатель также может использоваться для обмена сигналами. Некоторые электронные аналоговые телефоны имеют кнопку с надписью Link (Связь), при нажатии которой возникает *мгновенное событие сброса* (flash). Сброс можно выполнить вручную, нажав рычажный переключатель и удерживая его 200–1200 мс. Если удерживать переключатель в нажатом состоянии дольше, канал связи может интерпретировать это как завершение разговора. Назначение кнопки Link – реализация функциональности мгновенного нажатия рычажного переключателя. Если вы когда-нибудь использовали функцию отложенного вызова или вели разговор с подключением третьего абонента на аналоговой линии, вы применяли мгновенное нажатие рычажного переключателя с целью передачи сигналов в сеть.

**Телефонная трубка.** Телефонная трубка состоит из передатчика и приемника. Она осуществляет преобразования между звуковой энергией, используемой человеком, и электрической энергией, используемой телефонной сетью.

## Tip и Ring

В аналоговой телефонной сети имеется два провода. В Серверной Америке эти провода называют Tip (Земля) и Ring (Вход)[2]. Эта терминология сложилась во времена, когда соединение по телефону выполнялось живыми операторами на пультах коммутации. Используемые ими штекеры имели два контакта: один располагался на верхушке штекера, а другой подключался к кольцу, охватывающему штекер посередине (рис. 7.1).

Tip – это провод с положительной полярностью питания. В Северной Америке этот провод обычно зеленого цвета и обеспечивает обратную цепь. Ring – это провод с отрицательной полярностью. В Северной Америке этот провод обычно красный. Для современных кабелей Cat 5 и 6 Tip – это обычно белый провод, а Ring – цветной. Когда трубка те-

[1] Говоря о состоянии аналоговой линии, люди часто употребляют понятия «занято» и «свободно». Если линия «занята», значит, выполняется разговор по телефону. Если линия «свободна», телефон, по сути, выключен, или не занят.

[2] В других регионах их могут называть иначе (например, A и B).

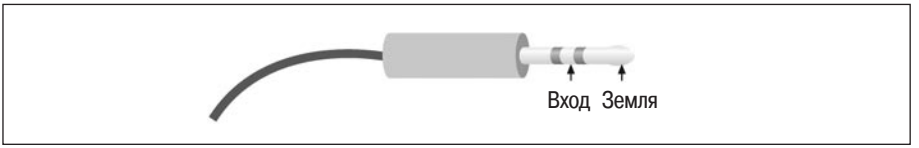



***Рис. 7.1.*** *Вход и Земля*

лефона не снята, напряжение на этом проводе относительно Tip – 48 В. При поднятии трубки это напряжение падает примерно до 7 В постоянного тока.

## Цифровая телефония

Аналоговая телефония уже практически мертва.

В PSTN единственный участок телефонной сети, до сих пор использующий технологию, которая появилась более ста лет назад, – это так называемая последняя миля[1] (The Last Mile).

Одна из основных сложностей при передаче аналоговых сигналов состоит в том, что помехой им может стать что угодно, приводя к снижению громкости, статическим помехам и всевозможным нежелательным эффектам. Вместо того чтобы оберегать аналоговый сигнал на больших расстояниях, которые могут иметь протяженность в тысячи километров, почему бы просто не измерить характеристики исходного звука и не послать на дальний конец эту информацию? Исходный сигнал, конечно, не дошел бы туда, но была бы передана вся информация, необходимая для его восстановления.

Таков принцип цифровой аудиосвязи (включая телефонию): измерение характеристик исходной звуковой волны, сохранение снятых показателей и передача этих данных на дальний конец. Затем на дальнем конце с помощью переданной информации формируется абсолютно новый аудиосигнал, обладающий теми же характеристиками, что и оригинальная звуковая волна. Воспроизведенная копия настолько хороша, что человеческое ухо не замечает разницы.

Принципиальное преимущество цифровой аудиосвязи заключается в возможности проверки оцифрованных данных математическими

[1] «Последняя миля» – термин, изначально используемый для описания единственного участка PSTN, не переведенного на оптоволоконные кабели: соединение между центральной АТС и абонентом. Однако последняя миля – это более широкое понятие, поскольку она имеет значение как ценный актив традиционных телефонных компаний; они владеют каналом связи, идущим в ваш дом. Последнюю милю становится все сложнее и сложнее описать с технической точки зрения, потому что сегодня существует огромное количество способов подключения абонента к сети. Если рассматривать ее как вещь, имеющую стратегическую ценность для телефонной компании, кабель и прочее оборудование, ее важность очевидна.

средствами на наличие ошибок на протяжении всего пути к месту назначения, что гарантирует поступление на дальний конец точной копии оригинала. Расстояние больше не влияет на качество, и помехи могут быть выявлены и устранены.

## Импульсно-кодовая модуляция

Существует несколько способов цифровой обработки аудиосигнала, но самым распространенным методом является импульсно-кодовая модуляция, или ИКМ (Pulse-Code Modulation, PCM). Чтобы проиллюстрировать принцип ее работы, рассмотрим несколько примеров.

### Цифровое кодирование аналогового сигнала

Принцип ИКМ заключается в измерении амплитуды[1] аналоговой волны через равные промежутки времени таким образом, чтобы позже ее можно было воссоздать. Количество замеров зависит как от разрядности квантования каждого замера, так и от частоты замеров. Чем больше разрядность и частота дискретизации, тем выше точность, но и тем бóльшая полоса пропускания потребуется для передачи такой подробной информации.

Чтобы лучше понять принцип работы ИКМ, рассмотрим волну, представленную на рис. 7.2.

Чтобы выполнить оцифровку волны, необходимо разбить ее на равные временные отрезки и замерить ее амплитуду в каждый момент време-

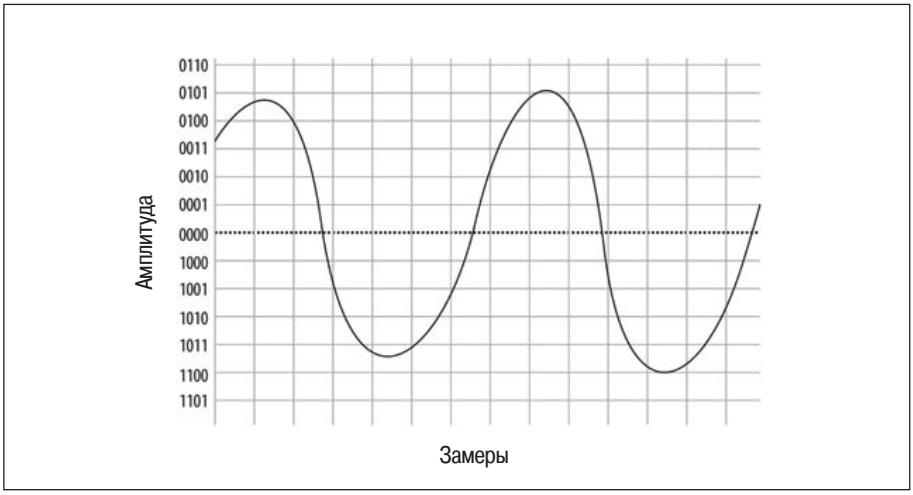



***Рис. 7.2.*** *Простая синусоидальная (гармоническая) волна*

[1] Амплитуда – это, по сути, мощность, или сила сигнала. Наверное, все когда-нибудь размахивали скакалкой или садовым шлангом, как кнутом, и видели получающуюся волну. Чем выше волна, тем больше ее амплитуда.

ни. Процесс разбиения волны на отрезки времени и измерение энергии в каждый момент называется *квантованием*, или *дискретизацией*.

Замеры должны производиться довольно часто, и необходимо собрать достаточно информации, чтобы обеспечить возможность восстановления волны на дальнем конце с приемлемой степенью точности. Для получения более точного замера потребуется большее количество битов. Чтобы объяснить этот принцип, начнем с очень маленького разрешения, при котором для представления амплитуды используется четыре бита. Это упростит задачу по визуализации и самого процесса квантования, и влияния, которое разрядность квантования имеет на качество.

На рис. 7.3 показано, какие данные будут зафиксированы, если выполнить дискретизацию гармонической волны с разрядностью четыре бита.

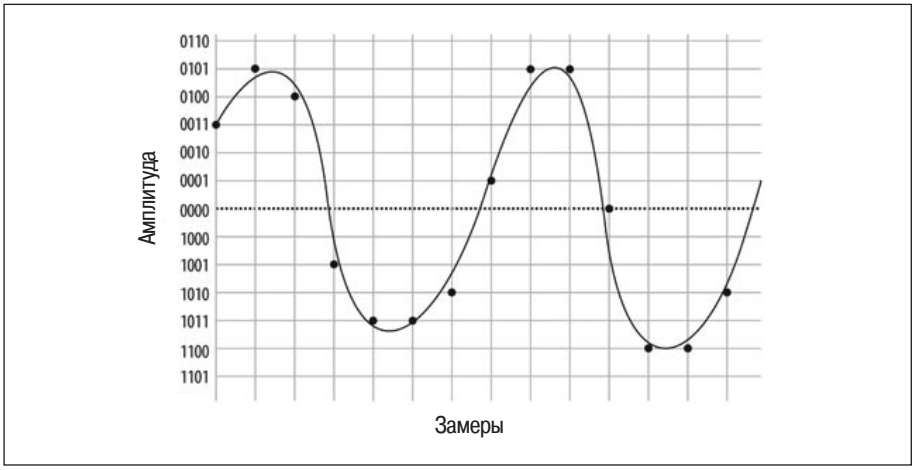



***Рис. 7.3.*** *Дискретизация гармонической волны с использованием четырех битов*

В каждом временном интервале измеряется амплитуда волны и записывается соответствующая интенсивность, иначе говоря, мы делаем замер. Как видите, разрядность в четыре бита ограничивает точность. Первый замер приходится округлять до 0011, следующий интервал дает значение 0101. Затем идут 0100, 1001, 1011 и т.д. В общем, получается 14 измерений (в реальности должно быть сделано несколько тысяч измерений в секунду).

Если из всех значений составить строку, их можно передавать на другой конец:

```
0011 0101 0100 1001 1011 1011 1010 0001 0101 0101 0000 1100 1100 1010
```

При передаче по проводам этот код выглядит примерно так, как показано на рис. 7.4.

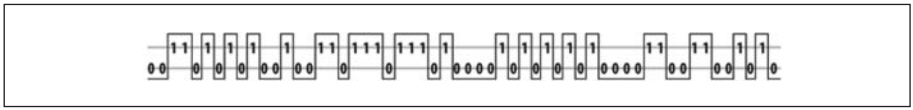


***Рис. 7.4.*** *ИКМ-кодированная волна*

Когда цифроаналоговый (digital-to-analog, D/A) преобразователь на дальнем конце получает этот сигнал, он может использовать данную информацию для построения волны, как показано на рис. 7.5.

На основании этих данных волна может быть восстановлена (рис. 7.6).

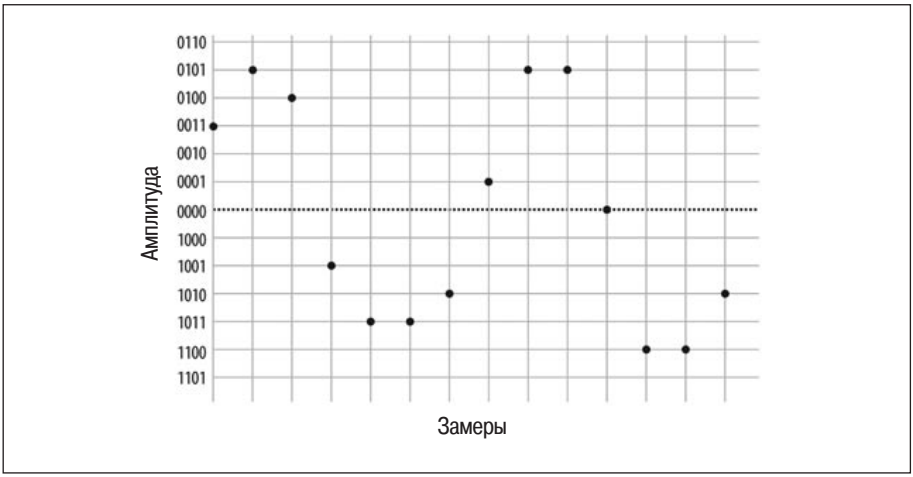



***Рис. 7.5.*** *Графическое представление ИКМ-сигнала*

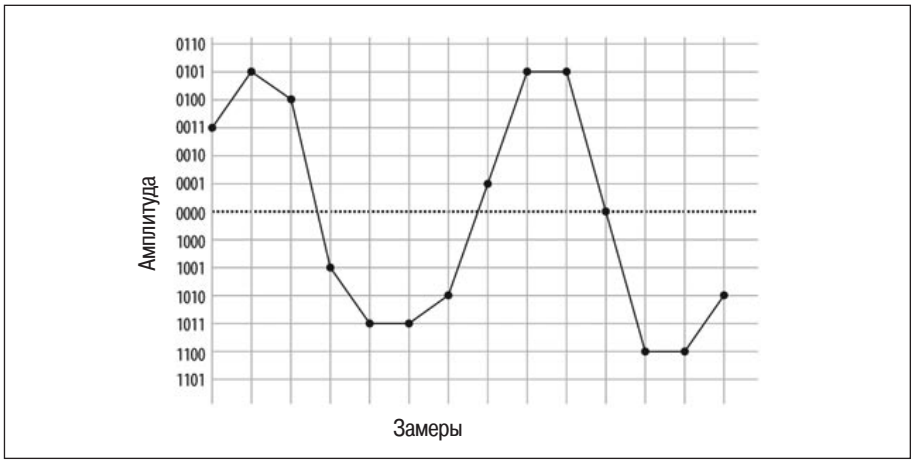



***Рис. 7.6.*** *Сигнал без сглаживания*

Как видите, если сравнить рис. 7.2 и 7.6, такая реконструкция волны не очень точная. Это было сделано намеренно, чтобы продемонстрировать важный момент: качество оцифрованной волны зависит от разрядности и частоты, с которой выполняются замеры. При слишком низкой частоте дискретизации качество получаемого аудиосигнала будет неприемлемым.

## Повышение разрешения и частоты дискретизации

Вернемся к исходной волне и на этот раз используем пять битов для определения интервалов квантования (рис. 7.7).

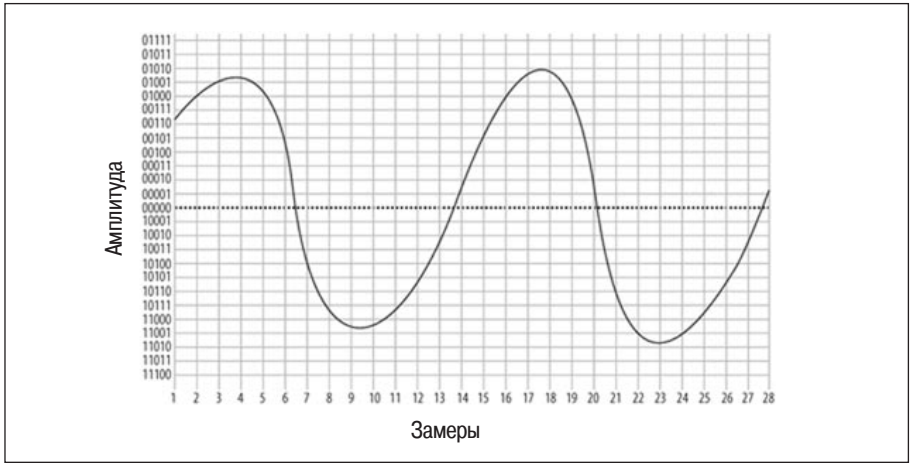



***Рис. 7.7.*** *Та же волна при более высокой разрядности квантования*

На самом деле пятибитовой ИКМ не существует. В телефонной сети замеры ИКМ кодируются с помощью 8 бит[1].

Также удвоим частоту дискретизации. Точки, откладываемые на этот раз, представлены на рис. 7.8.

Теперь количество замеров и разрядность увеличены вдвое. Вот полученные данные:

```
00111 01000 01001 01001 01000 00101 10110 11000 11001 11001 11000 10111 10100
10001 00010 00111 01001 01010 01001 00111 00000 11000 11010 11010 11001 11000
10110 10001
```

При получении на другом конце эти данные могут быть представлены так, как показано на рис. 7.9.

На основании этой информации может быть построена волна, представленная на рис. 7.10.

[1] Другие методы цифровой аудиозаписи могут использовать 16 бит или более.

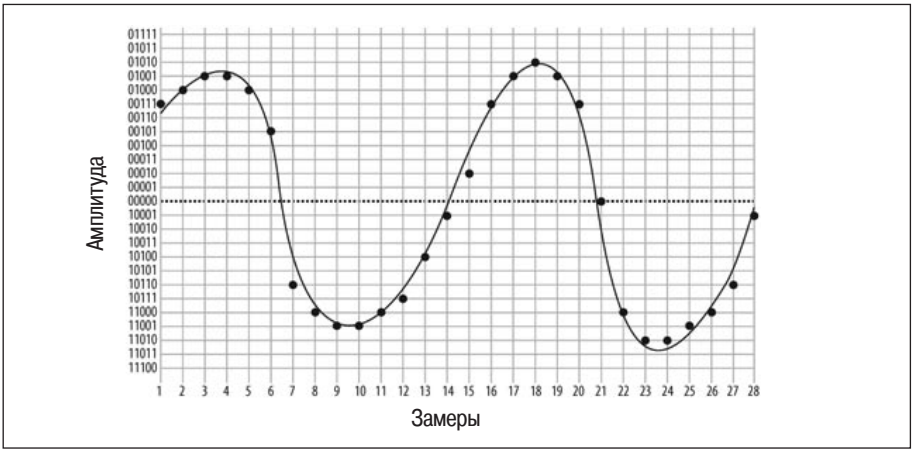



***Рис. 7.8.*** *Та же волна при вдвое большей разрядности*

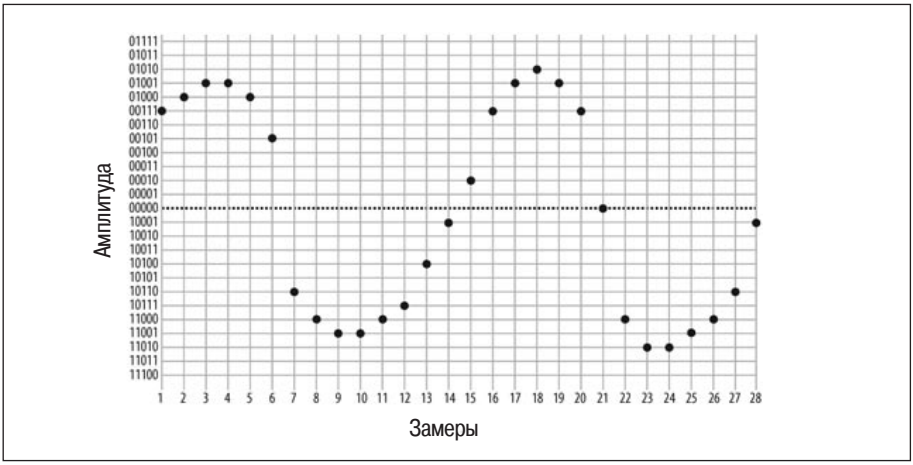



***Рис. 7.9.*** *ИКМ-сигнал с разрядностью 5 бит*

Как видите, полученная в данном случае волна намного более точно представляет оригинал. Однако также можно заметить, что все равно имеется возможность для улучшения.

Обратите внимание, что при кодировании волны с разрядностью квантования 4 бита использовалось 40 бит, тогда как для отправки той же волны с разрядностью квантования 5 бит (и также вдвое большей частотой дискретизации) пришлось использовать 156 бит. Суть в том, что существует соотношение: чем лучшее качество необходимо обеспечить при кодировании аудиосигнала, тем больше битов для этого используется, и тем больше битов придется передавать (естественно, в реальном масштабе времени), и тем бóльшая полоса пропускания потребуется.

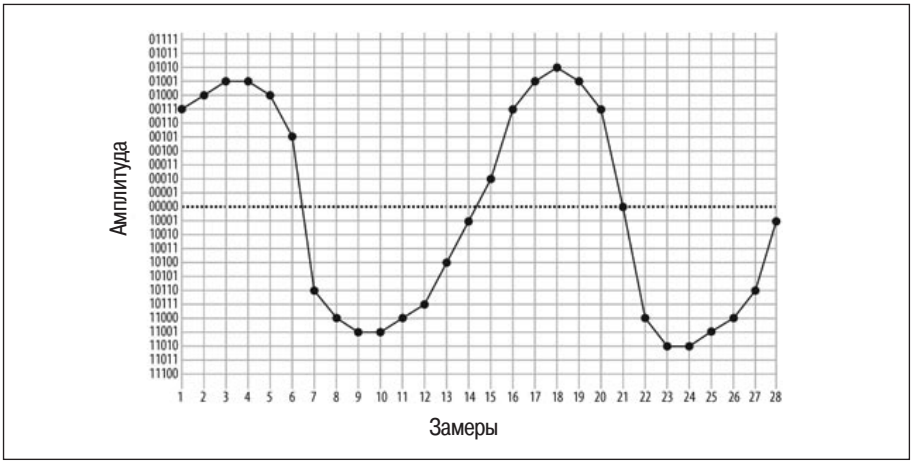



***Рис. 7.10.*** *Волна, полученная из ИКМ-сигнала с разрядностью 5 бит*

## Теорема Найквиста

Итак, какая частота дискретизации будет достаточной? Такой же вопрос задал себе в 20-х годах прошлого века инженер-электрик (и сотрудник AT&T/Bell) Гарри Найквист (Harry Nyquist). Теорема Найквиста гласит: «Чтобы можно было абсолютно точно восстановить исходный сигнал по его дискретной версии, частота дискретизации должна быть вдвое больше полосы частот входного сигнала»[1].

По существу, это означает следующее: чтобы точно кодировать аналоговый сигнал, частота замеров должна вдвое превышать максимальную частоту в частотном диапазоне, который требуется воспроизвести. Поскольку телефонная сеть не будет передавать частоты ниже 300 и выше 4000 Гц, частоты дискретизации 8000 замеров в секунду будет достаточно для воспроизведения любой частоты в частотном диапазоне аналогового телефона. Запомните величину 8000 замеров в секунду; мы поговорим об этом немного позже.

## Компандирование по логарифмическому закону

Итак, мы рассмотрели основы квантования и обсудили тот факт, что большее количество интервалов квантования (то есть более высокая

[1] Найквист опубликовал две статьи, «Certain Factors Affecting Telegraph Speed» (1924) и «Certain Topics in Telegraph Transmission Theory» (1928), в которых постулировал свою теорему, ставшую известной как теорема Найквиста. Подтвержденная в 1949 году Клодом Шенноном (Claude Shannon) («Communication in the Presence of Noise»), она получила название «теорема о дискретном представлении Найквиста–Шеннона».

частота дискретизации) обеспечивают лучшее качество, но при этом требуется бóльшая полоса пропускания. Наконец, мы обсудили, какой должна быть минимальная частота дискретизации для точного измерения диапазона частот, который мы хотим передавать (в случае с телефоном это 8000 Гц). Все это сводится к тому, что по проводам передается довольно большое количество данных, поэтому пришла пора поговорить о компандировании.

*Компандирование* – это метод улучшения динамического диапазона метода дискретизации без потери необходимого качества. Он заключается в квантовании более высоких амплитуд с намного меньшей частотой, чем меньших амплитуд. Иначе говоря, если кричать в телефон, ваш голос будет дискретизирован не так хорошо, как если бы вы говорили нормально. Крик также вреден для кровяного давления, поэтому лучше избегать его.

Обычно используется два метода компандирования: μlaw[1] в Северной Америке и alaw в остальном мире. В них используется один принцип, но во всем остальном они несовместимы.

При компандировании волна делится на *хорды*, каждая из которых включает несколько *шагов*. Квантование заключается в сопоставлении измеренной амплитуды с соответствующим шагом хорды. Значение полосы и номера хорд (а также знак – плюс или минус) становятся сигналом. Приведенные далее диаграммы дают визуальное представление того, что происходит при компандировании. Они не основываются на каком-либо стандарте, а просто созданы как иллюстрация (опять же, в телефонной сети компандирование будет выполняться с разрядностью 8, а не 5 бит).

Рис. 7.11 иллюстрирует компандирование с разрядностью 5 бит. Как видите, дискретизация амплитуд, близких к нулевому уровню, намного выше, чем больших амплитуд (как в положительной, так и в отрицательной области). Однако, поскольку человеческое ухо, передатчик и приемник также будут искажать громкие сигналы, это не представляет проблемы.

Квантованный образец может выглядеть так, как представлено на рис. 7.12. Он обеспечивает получение следующего потока битов:

```
00000 10011 10100 10101 01101 00001 00011 11010 00010 00001 01000 10011 10100
10100 00101 00100 00101 10101 10011 10001 00011 00001 00000 10100 10010 10101
01101 10100 00101 11010 00100 00000 01000
```

[1] μlaw часто называют ulaw, потому что – посмотрим правде в глаза – у кого из нас есть клавиша μ на клавиатуре? μ – это фактически греческая буква мю; поэтому также можно встретить выражение (более точное) «Mu-law».

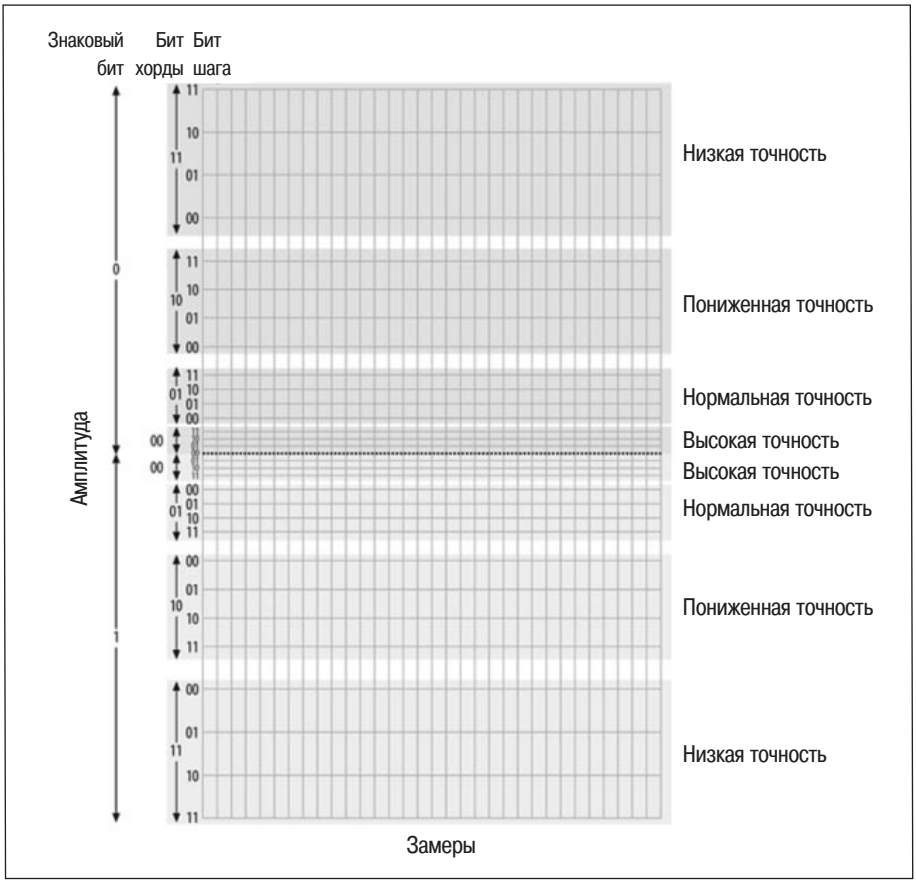



***Рис. 7.11.*** *Компандирование с разрядностью 5 бит*

## Наложение частот

Если когда-нибудь при просмотре одного из старых вестернов вам казалось, что колеса повозки вращаются в обратном направлении, вы наблюдали эффект наложения частот. Частота смены кадров фильма не соответствует частоте вращения колес, и поэтому возникает впечатление вращения в обратную сторону.

В цифровой аудиосистеме (каковой является современная PSTN) наложение частот возникает, если в аналогово-цифровой преобразователь поступают сигналы частотой, превышающей половину частоты дискретизации. В PSTN это аудиосигналы частотой выше 4000 Гц (половина частоты дискретизации, которая составляет 8000 Гц). Эту проблему легко исправить, пропустив аудиосигнал через низкочастотный фильтр[1] перед его передачей в аналогово-цифровой преобразователь[2].

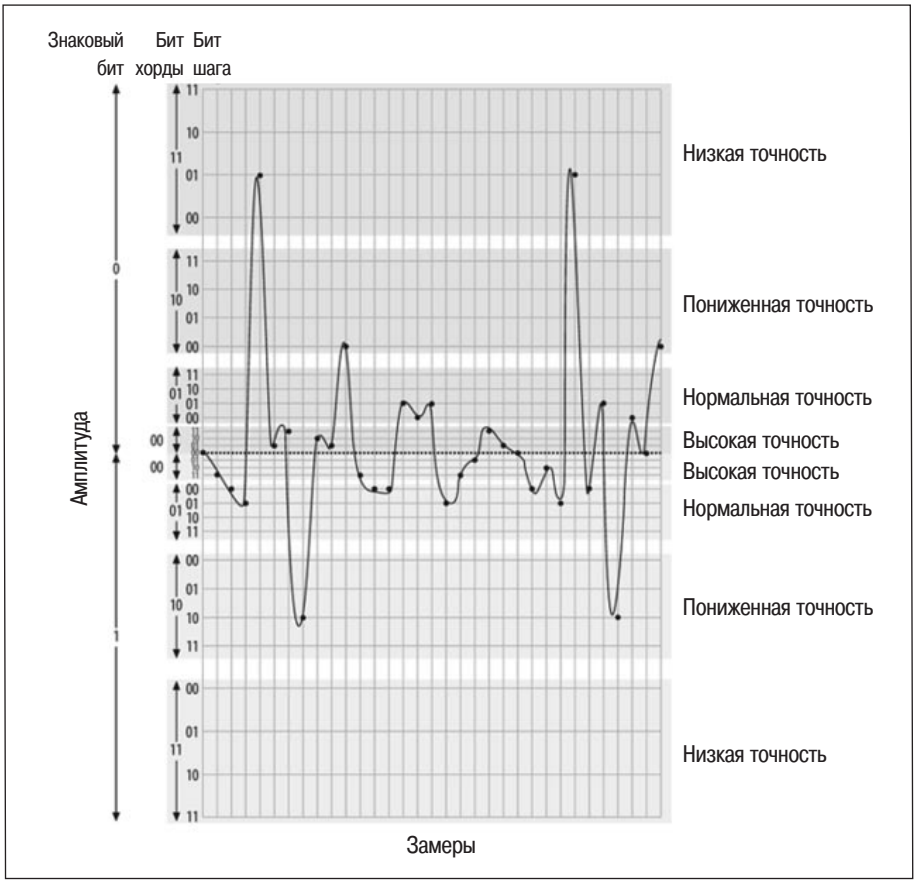



***Рис. 7.12.*** *Квантование и компандирование с разрядностью 5 бит*

---

[1] Низкочастотный фильтр, как следует из его названия, пропускает только частоты, которые меньше его частоты среза. Существуют еще высокочастотные фильтры (которые убирают низкие частоты) и полосные фильтры (которые отфильтровывают и высокие, и низкие частоты).

[2] Те, кому когда-либо приходилось делать аудиозаписи для системы, вероятно, пользовались полосным фильтром, который встроен в большинство телефонных аппаратов. Запись с использованием даже высококлассного звукозаписывающего оборудования может выявить разнообразнейший фоновый шум. Вы могли даже не слышать его, пока не выполнили понижающей дискретизации, при которой фоновый шум создает эффект наложения частот (что может порождать разнообразные странные звуки). С другой стороны, телефонные записи уже выполнены в соответствующем формате, поэтому шум никогда не попадает в аудиопоток. Из всего вышесказанного следует, что, независимо от того, какое оборудование используется для записи, следует избегать сред с большим количеством фоновых шумов. Обычные офисы могут быть намного более шумными, чем кажется, поскольку оборудование отопления, вентиляции и кондиционирования воздуха может создавать шум, о котором мы даже не подозреваем.

# Цифровая коммутируемая телефонная сеть

Более ста лет телефонные сети были исключительно коммутируемыми. Это означало, что для каждого телефонного звонка между двумя конечными точками устанавливалось выделенное соединение с фиксированной полосой частот, распределенной для этого канала. Создание такой сети было дорогим удовольствием, а использование ее на больших расстояниях обходилось вдвое дороже. Хотя все предсказывают окончание эпохи коммутируемой сети, многие люди по-прежнему пользуются ею каждый день и она, действительно, достаточно неплохо справляется со своими функциями.

## Типы линий связи

В PSTN существует много разных линий, обеспечивающих различные нужды сети. Между центральной АТС и абонентом обычно достаточно одной или более аналоговых линий или нескольких десятков каналов, предоставляемых посредством цифровой линии. Между станциями PSTN (и большим количеством абонентов), как правило, используются оптоволоконные линии связи.

### Простая DS-0 – основа всего

Поскольку стандартный метод оцифровки телефонного звонка – запись 8-битового замера 8000 раз в секунду, можно заметить, что телефонной линии с ИКМ понадобится полоса пропускания 8000 бит/с × 8, то есть 64 000 бит/с. Такой канал с полосой пропускания 64 Кбит/с называют DS-0. DS-0 – это основной строительный блок всех цифровых телекоммуникационных линий.

Даже вездесущая аналоговая линия переходит на DS-0 ускоренными темпами. Иногда это происходит в месте, где линия входит в центральную АТС, иногда намного раньше[1].

### Линии с T-несущей

T1 – один из самых известных терминов цифровой телефонии. T1 – это цифровая линия, состоящая из 24 мультиплексирующихся каналов DS-0, обеспечивающих передачу данных со скоростью 1,544 Мбит/с[2]. Этот битовый поток определен как DS-1. Голос кодируется в T1 с использованием алгоритма компандирования μlaw.

---

[1] Цифровые телефоны (включая IP-телефоны) выполняют аналого-цифровое преобразование непосредственно в точке подключения телефонной трубки к телефону; таким образом, DS-0 берет начало прямо в телефонном аппарате.

[2] 24 канала DS-0 используют 1,536 Мбит/с, оставшиеся 0,008 Мбит/с используются битами синхронизации.

Европейская версия T1 была разработана Европейской конференцией почтовых и телекоммуникационных ведомств (European Conference of Postal and Telecommunications Administrations, CEPT) и сначала называлась CEPT-1. Теперь ее называют E1.

E1 образована 32 каналами DS-0, но метод ИКМ-кодирования другой: E1 использует закон компандирования alaw. Это означает, что для соединения между сетями E1 и T1 всегда будет необходим этап перекодировки. Заметьте, что E1, хотя и имеет 32 канала, также считается линией DS-1. E1 намного больше распространена, поскольку используется во всем мире, кроме Северной Америки и Японии.

Все остальные линии с T-несущей (T2, T3 и T4) являются кратными T1 и базируются на простой DS-0. В табл. 7.2 представлены сравнительные характеристики разных линий с T-несущей.

*Таблица 7.2. Линии с T-несущей*

| Несущая | Эквивалент по скорости передачи данных | Количество каналов DS-0 | Скорость передачи данных, Мбит/с |
|---|---|---|---|
| T1 | 24 канала DS-0 | 24 | 1,544 |
| T2 | 4 канала T1 | 96 | 6,312 |
| T3 | 7 каналов T2 | 672 | 44,736 |
| T4 | 6 каналов T3 | 4032 | 274,176 |

При плотностях более T3 линии с T-несущей используются очень редко. Для таких скоростей передачи данных могут применяться оптоволоконные линии связи (Optical Carrier, OC).

## Синхронная оптическая сеть и оптоволоконные линии связи

Синхронная оптическая сеть (Synchronous Optical Network, SONET) была разработана по причине необходимости перевода системы с T-несущей на следующий технологический уровень – волоконную оптику. SONET базируется на полосе пропускания T3 (44,736 Мбит/с) с небольшими потерями, что в сумме составляет 51,84 Мбит/с. Такая линия называется OC-1 или STS-1. Как показано в табл. 7.3, скорость передачи данных во всех высокоскоростных оптоволоконных линиях кратна этой базовой величине.

Создание SONET было попыткой стандартизации оптических линий. Однако ее высокая стоимость, а также преимущества, предлагаемые более новыми схемами, такими как мультиплексирование по длине волны высокой плотности (Dense Wave Division Multiplexing, DWDM), делают ее будущее туманным.

*Таблица 7.3. Оптоволоконные линии связи*

| Несущая | Эквивалент по скорости передачи данных | Количество каналов DS-0 | Скорость передачи данных, Мбит/с |
|---|---|---|---|
| OC-1 | 1 канал DS-3 (плюс полоса на непроизводительные затраты) | 672 | 51,840 |
| OC-3 | 3 канала DS-3 | 2016 | 155,520 |
| OC-12 | 12 каналов DS-3 | 8064 | 622,080 |
| OC-48 | 48 каналов DS-3 | 32256 | 2488,320 |
| OC-192 | 192 канала DS-3 | 129024 | 9953,280 |

## Протоколы обмена сигналами по цифровым каналам

Как для всех остальных линии, для линий, используемых в PSTN, недостаточно просто передавать данные (голос) между конечными точками. Должны существовать механизмы для обмена информацией о статусе канала. (Контроль за разъединением и ответом – вот два основных примера использования обмена сигналами; Caller ID (ID звонящего) – пример более сложной формы обмена сигналами.)

### Передача служебных сигналов по выделенному каналу

Метод передачи служебных сигналов по выделенному каналу (Channel Associated Signaling, CAS), называемый также обменом сигналов с резервированием битов, используется для передачи голоса по линии T1, если недоступна ISDN. CAS не использует мощь цифровой линии, а имитирует аналоговые каналы. CAS обеспечивается за счет заимствования битов из аудиопотока для служебных целей. Хотя это не оказывает заметного влияния на качество передаваемого аудиосигнала, отсутствие мощного канала для передачи служебных сигналов негативно влияет на гибкость.

При конфигурации линии T1 с CAS опции обмена сигналами на каждом конце должны совпадать. Как правило, предпочтительной является технология обмена сигналами E&M (Ear & Mouth или recEive & transMit), поскольку она предлагает наилучший контроль. Исходя из сказанного наиболее вероятное применение CAS в среде Asterisk – банк каналов. Это означает, что, скорее всего, вам придется использовать протокол обмена сигналами FXS.

Теперь CAS очень редко используется в линиях PSTN из-за преимуществ, предлагаемых ISDN-PRI. Одно из ограничений CAS – он не допускает динамического назначения функций каналам. Также информацию Caller ID (такая функция может даже не поддерживаться) приходится передавать как часть аудиопотока. CAS обычно используется в линии T1 в банках каналов.

## ISDN

Цифровая сеть с интеграцией служб (Integrated Services Digital Network, ISDN) появилась более 20 лет назад. Поскольку она разделяет каналы, по которым передается основной трафик (несущие или B-каналы), и канал, переносящий сигнальную информацию (D-канал), ISDN обеспечивает возможность предоставления намного более богатой функциональности, чем CAS. Вначале ожидалось, что ISDN будет обеспечивать возможности, сходные с теми, которые нам дает Интернет, включая улучшенные средства передачи голоса, видео и данных.

К сожалению, вместо того чтобы ратифицировать стандарт и придерживаться его, каждый уважающий себя производитель средств связи решил внести в протокол что-то свое, твердо веря, что его версия является самой лучшей и в конце концов займет лидирующее положение на рынке. В результате соединение двух ISDN-совместимых систем часто становилось болезненной и дорогостоящей процедурой. Поставщики услуг связи, которые должны были реализовать и поддерживать эту дорогостоящую технологию, в свою очередь, установили непомерно высокие цены на нее, что затормозило ее распространение. В настоящее время ISDN используется практически исключительно для базового объединения каналов. Кстати, аббревиатуру ISDN в отрасли в шутку расшифровывают как «It Still Does Nothing» (Она до сих пор ничего не делает).

Таким образом, основным применением ISDN стало объединение каналов, и сейчас она (в основном) соответствует стандартам. Если имеется офисная АТС с более чем десятком линий, подключенных к PSTN, скорее всего, будет использоваться линия ISDN-PRI (Primary Rate Interface). Также там, где нет возможности доступа к Интернету по цифровой абонентской линии (Digital Subscriber Line, DSL) или кабельной линии (или он очень дорогой), канал ISDN-BRI (Basic Rate Interface) мог бы обеспечить доступное по цене соединение со скоростью передачи данных 128 Кбит/с. В большей части Северной Америки отказались от использования BRI для соединения с Интернетом в пользу DSL и кабельных модемов (и ее никогда не применяли для передачи голоса), но во многих европейских странах она практически полностью заменила аналоговые линии.

**ISDN-BRI/BRA.** Интерфейс, обеспечивающий базовую скорость передачи данных (Basic Rate Interface) или базовый доступ (Basic Rate Access) – разновидность ISDN, разработанная для обслуживания конечных точек с малой нагрузкой на канал, таких как рабочие станции.

Разновидность BRI спецификации ISDN часто называют просто ISDN, но это может стать причиной путаницы, поскольку ISDN – это протокол, а не тип линии (не говоря уже о том, что линии PRI также правильно было бы называть ISDN!).

Линия Basic Rate ISDN состоит из двух B-каналов со скоростью передачи данных 64 Кбит/с, управляемых D-каналом со скоростью передачи данных 16 Кбит/с, что в сумме составляет 144 Кбит/с.

Линия Basic Rate ISDN является источником неразберихи на протяжении всего времени своего существования из-за проблем совместимости со стандартами, сложности с технической точки зрения и недостатка документации по ней. Тем не менее многие европейские телефонные компании широко применяют ISDN-BRI, таким образом, она более популярна в Европе, чем в Северной Америке.

**ISDN-PRI/PRA.** Интерфейс, обеспечивающий основную скорость передачи данных (Primary Rate Interface) или основной доступ (Primary Rate Access) – разновидность ISDN, используемая для предоставления сервиса ISDN через более масштабные сетевые соединения. Линия Primary Rate ISDN использует для передачи служебных сигналов один канал DS-0 (D-канал); оставшиеся каналы выполняют роль B-каналов.

В Северной Америке линии Primary Rate ISDN обычно строятся на одном или более T1-каналах. Поскольку T1 включает 24 канала, североамериканская PRI-линия обычно состоит из 23 B-каналов и одного D-канала. Поэтому PRI-линии часто обозначают как 23B+D[1].

В Европе используется 32-канальная линия E1, поэтому линия Primary Rate ISDN обозначается как 30B+D (последний канал используется для синхронизации).

Линии Primary Rate ISDN очень популярны благодаря техническим преимуществам и, как правило, конкурентоспособной цене при более высоких плотностях. Если планируется использовать примерно десяток PSTN-линий или более, следует обратить внимание на цену Primary Rate ISDN.

С технической точки зрения ISDN-PRI всегда предпочтительнее CAS.

## Signaling System 7

Signaling System 7 (SS7) – это система обмена сигналами, используемая поставщиками услуг связи. Концептуально она аналогична ISDN и обеспечивает поставщикам эффективный механизм передачи дополнительной информации, которую обычно должны передавать конеч-

[1] PRI на самом деле обладают намного большей гибкостью, поскольку одна PRI-линия может объединять несколько каналов T1. Таким образом можно получить линию 47B+D (где один D-канал обслуживает два канала T1) или 46B+2D (где основной и резервный D-каналы обслуживают два канала T1). Иногда можно увидеть PRI-линию, описанную как nB+nD, потому что количество B- и D-каналов на самом деле может быть разным. Поэтому никогда не следует называть канал T1, использующий PRI, просто PRI. К вашему сведению, линия PRI, объединяющая несколько каналов T1, – обычное явление в больших офисных АТС.

ные точки ISDN. Однако технология SS7 отличается от ISDN. Главное различие в том, что SS7 выполняется в совершенно отдельной сети, независимо от магистральных линий связи, по которым осуществляются звонки.

Реализация поддержки SS7 в Asterisk не за горами, поскольку очень велик интерес сделать Asterisk совместимой с сетями поставщиков услуг телефонной связи. Версия SS7 с открытым исходным кодом (*http://www.openss7.org*) существует, но еще требует доработки для полной совместимости с SS7. На момент написания данной книги неизвестно, будет ли эта версия интегрирована с Asterisk. Другой многообещающий источник поддержки SS7 обеспечивает компания Sangoma Technologies, которая предлагает функциональность SS7 во многих своих продуктах.

Следует отметить, что введение поддержки SS7 в Asterisk не заключается просто в написании соответствующего драйвера. Подключение оборудования к SS7-сети будет невозможным без прохождения этим оборудованием исключительно жесткой сертификации. И даже после этого вряд ли кто-нибудь из поставщиков традиционных услуг связи поспешит обеспечить условия для того, чтобы это произошло, главным образом, из стратегических и политических соображений.

## Сети с коммутацией пакетов

В середине 1990-х годов производительность сетей достигла той точки, когда стало возможно передавать через сетевые соединения поток медиа-информации в режиме реального времени. Поскольку медиа-поток разделяется на сегменты, заключенные в конверт с адресом, такие соединения называют *пакетными*. Основная сложность заключается, конечно, в том, чтобы переслать множество таких пакетов между двумя конечными точками, обеспечив их поступление в том же порядке, в каком они были отправлены, менее чем за 150 мс и без потерь. В этом суть технологии передачи голоса по IP-протоколу.

## Заключение

В данной главе были рассмотрены технологии, используемые в настоящее время в PSTN. В следующей главе мы обсудим протоколы для VoIP: передачу телефонных соединений по сетям, использующим протоколы IP. Данные протоколы определяют разные механизмы для осуществления телефонных разговоров, но их значимость этим не ограничивается. Вход телефонной сети в сеть передачи данных окончательно устранит барьер между телефонами и компьютерами, что обещает привести к революционным изменениям в способах общения.

# 8

# Протоколы для VoIP

*Интернет – это зазнавшаяся система телефонной связи.*
– Клиффорд Столл

Телекоммуникационная промышленность существует более 100 лет, и Asterisk объединяет в себе большинство, если не все основные технологии, применявшиеся в это последнее столетие. Чтобы максимально эффективно работать с Asterisk, не надо быть профессионалом во всех областях, но понимание разницы между разнообразными кодеками и протоколами обеспечит лучшее восприятие и осмысление системы в целом.

В данной главе рассматривается технология передачи голоса по IP-протоколу (Voice over IP, VoIP) и то, что отличает сети VoIP от традиционных коммутируемых телефонных сетей, которые были темой предыдущей главы. Мы исследуем, зачем нужны протоколы VoIP, кратко коснувшись истории и возможного будущего для каждого из них. Также мы обратим внимание на вопросы безопасности и способность этих протоколов работать в сетях, использующих технологию трансляции сетевых адресов (Network Address Translation, NAT). Остановимся на следующих протоколах VoIP (некоторые из них будут рассмотрены менее подробно, чем другие):

- IAX
- SIP
- H.323
- MGCP
- Skinny/SCCP
- UNISTIM

Кодеки – это средства, с помощью которых аналоговый голосовой сигнал может быть преобразован в цифровой сигнал и передан по Интернету. Пропускная способность любого устройства ограничена, и количество одновременных разговоров, которое может обеспечивать любое отдельно взятое соединение, напрямую зависит от типа используемого кодека. В данной главе мы также рассмотрим, чем отличаются следующие кодеки с точки зрения требований к полосе пропускания (уровень сжатия) и качества:

- G.711
- G.726
- G.729A
- GSM
- iLBC
- Speex
- MP3

Глава завершится обсуждением возможностей надежной передачи голосового трафика, причин возникновения эха и средств борьбы с ним, а также того, как Asterisk управляет аутентификацией входящих и исходящих звонков.

## Зачем нужны протоколы VoIP

Основная предпосылка использования VoIP – пакетирование[1] аудиопотоков для транспортировки по сетям, использующим протокол IP (Internet Protocol). Главные сложности при этом заключаются в манере общения людей. Сигнал должен не только поступить практически в той же форме, в какой был передан, но его транспортировка должна занять не более 150 мс. Если пакеты будут утеряны или задержатся, качество связи ухудшится, то есть два человека будут испытывать трудности в ведении беседы.

Транспортные протоколы, которые объединены под общим названием «сетевые», изначально разрабатывались без реализации возможности потоковой передачи несущей в режиме реального времени. Предполагалось, что конечные точки в случае потери пакетов будут увеличивать время их ожидания, посылая запросы на повторную передачу или в некоторых случаях просто продолжать работать без потерянной информации. Для обычного голосового общения такие механизмы неприемлемы. Наши разговоры не допускают утраты букв или слов и тем более какой-либо ощутимой задержки между передачей и приемом.

[1] Это слово не вполне литературное, но данный термин приобретает все большее и большее распространение. Он означает процесс разделения непрерывного потока информации на фрагменты (или пакеты), которые могут доставляться независимо друг от друга.

Традиционная PSTN была разработана специально для передачи голоса и прекрасно подходит для выполнения этой задачи с технической точки зрения. Однако с точки зрения гибкости ее недостатки очевидны даже тем, кто слабо разбирается в этой технологии. VoIP обещает включить телефонную связь во все другие протоколы, используемые в наших сетях, но из-за особых требований к передаче разговоров для разработки, создания и обслуживания таких сетей необходимо обладать специальными навыками.

Проблема с пакетной передачей голоса заключается в том, что то, как мы говорим, абсолютно не совпадает с тем, как IP передает данные.

Процесс разговора и слушания состоит из ретрансляции потока аудиосигналов, тогда как сетевые протоколы разработаны так, что они все разбивают на части, заключают единицы информации в тысячи пакетов и затем доставляют каждый пакет на дальний конец линии связи любым возможным путем. Очевидно, с этим надо что-то делать.

# Протоколы VoIP

Механизм соединения по протоколу VoIP обычно состоит в сериях транзакций по передаче сигналов между конечными точками (и шлюзами, располагающимися между ними), которые оформляются в два устойчивых медиа-потока (по одному в каждом направлении), фактически передающих беседу. Есть несколько протоколов для осуществления этого. В данном разделе мы обсудим те из них, которые имеют значение для VoIP в общем и Asterisk в частности.

## IAX (протокол Inter-Asterisk eXchange)

Чтобы проверить человека, который заявляет, что является знатоком Asterisk, попросите его прочитать название этого протокола. Казалось бы, оно должно звучать, как «ай-эй-экс», но так можно и язык сломать[1]. К счастью, правильно он произносится «иикс»[2]. IAX – это открытый протокол, то есть кто угодно может загружать его и вести его разработку, но он пока что не является стандартом[3].

Ожидается, что IAX2 вскоре станет IETF-протоколом. В настоящее время IAX2 находится в IETF в статусе проекта и ожидается, что он станет официальным протоколом в течение нескольких лет.

В Asterisk поддержку IAX обеспечивает модуль chan_iax2.so.

---

[1] Звучит как название голландской футбольной команды.

[2] Ну, давайте, произнесите вслух. Теперь звучит намного лучше, не так ли?

[3] Официально текущей версией этого протокола является IAX2, но, поскольку от всякой поддержки IAX1 отказались, под IAX и IAX2 подразумевается одна и та же версия.

## История

Протокол IAX был разработан компанией Digium для обмена информацией с другими серверами Asterisk (отсюда и название: протокол Inter-Asterisk eXchange). Крайне важно отметить, что IAX не ограничен применением только в Asterisk. Этот стандарт открыт для использования и поддерживается многими телекоммуникационными проектами с открытым исходным кодом, а также несколькими производителями оборудования. IAX – это транспортный протокол (подобно SIP), который использует один порт UDP (4569) и для обмена сигналами по каналам, и для медиа-потоков. Как объясняется ниже в данной главе, это упрощает обслуживание при использовании межсетевых экранов, поддерживающих NAT.

IAX также обладает уникальной способностью объединять несколько сеансов в один поток данных, что может обеспечивать громадный выигрыш по пропускной способности при отправке множества одновременных каналов на удаленный сервер. Объединение позволяет представлять множество медиа-потоков под одним заголовком датаграммы (datagram), что сокращает издержки на отправку каждого отдельно взятого канала. Таким образом снижаются задержки и сокращаются требования к вычислительной мощности и пропускной способности, что обеспечивает возможность масштабирования протокола для поддержания большого числа активных каналов между конечными точками. Если требуется передавать большое количество IP-вызовов между двумя конечными точками, следует обратить особое внимание на способность IAX объединять каналы связи.

## Будущее

Поскольку IAX был оптимизирован для передачи голоса, его критикуют за недостаточную поддержку видео, но на самом деле потенциально IAX может передавать практически любой медиа-поток. Поскольку это открытый протокол, в него, несомненно, будет включена возможность передачи любых типов медиа-данных, которые появятся в будущем, если сообществу это понадобится.

## Вопросы безопасности

IAX включает возможность аутентификации тремя способами: открытый текст, хеширование MD5 и обмен ключами RSA. Конечно, это никак не касается шифрования медиа-потоков или заголовков при передаче между конечными точками. Многие решения включают использование устройства или программного обеспечения виртуальной частной сети (Virtual Private Network, VPN) для шифрования потока на другом уровне технологии, при котором от конечных точек требуется заранее установить правила, по которым будут конфигурироваться и работать эти каналы. Однако сейчас IAX также может шифровать потоки между конечными точками с использованием динамического об-

мена ключами при установлении соединения (за счет применения конфигурационной опции `encryption=aes128`), что обеспечивает возможность использования автоматического выбора ключей.

### IAX и NAT

Протокол IAX2 был специально разработан для работы с устройствами, находящимися за межсетевыми экранами, которые реализуют протокол NAT. Использование одного UDP-порта и для обмена служебными сигналами, и для передачи голоса также сводит к минимуму количество каналов, которые необходимо открыть в межсетевом экране. Эти условия помогли сделать IAX одним из простейших протоколов (если не самым простым) для реализации в безопасных сетях.

## SIP

Протокол Session Initiation Protocol (SIP) покорил телекоммуникационную отрасль. SIP практически низверг с пьедестала когда-то могущественный H.323 и стал предпочтительным протоколом VoIP, безусловно, в конечных точках сети. Основная идея SIP в том, что каждый конец соединения является равноправным участником сети; протокол договаривается о параметрах устанавливаемого между ними соединения. Неотразимым протокол SIP делает его относительная простота; его синтаксис подобен синтаксису многих широко известных протоколов, таких как HTTP и SMTP. Поддержку SIP в Asterisk обеспечивает модуль chan_sip.so[1].

### История

Впервые SIP был представлен в организации Internet Engineering Task Force (IETF) в феврале 1996 года как draft-ietf-mmusic-sip-00. Исходный проект не имел ничего общего с тем, каким мы знаем SIP сегодня, и содержал только один тип запросов: запрос на установление соединения. В марте 1999 года, после 11 редакций, родился SIP RFC 2543.

Поначалу SIP практически полностью игнорировался, поскольку H.323 считался предпочтительным протоколом для определения параметров соединения при транспортировке данных по VoIP. Однако по мере нарастания шума вокруг него SIP начал завоевывать популярность. И хотя, возможно, его развитие ускорили различные факторы, нам приятнее думать, что в большей мере его успех обусловлен свободно доступной спецификацией.

---

[1] После того как мы только что назвали SIP простым, следует отметить, что он ни в коем случае не является примитивным. Если бы кто-то решил прочитать все RFC IETF, касающиеся SIP, ему пришлось бы осилить более 3000 страниц. SIP быстро приобретает репутацию слишком раздутого протокола, но это никак не умаляет его популярности.

SIP – это сигнальный протокол уровня приложений, который использует для передачи информации широко известный порт 5060. SIP-пакеты могут передаваться по протоколам транспортного уровня UDP или TCP. В настоящее время Asterisk не имеет реализации TCP для передачи SIP-сообщений, но возможно, будущие версии будут поддерживать его (приветствуются патчи к кодовой базе). SIP используется для «установления, корректировки и завершения сеансов обмена мультимедийной информацией, таких как звонки интернет-телефонии»[1].

SIP не передает речевые данные между конечными точками.

Для передачи речевых данных (то есть голоса) между конечными точками применяется RTP. RTP использует в Asterisk непривилегированные порты с большими порядковыми номерами (по умолчанию от 10000 до 20000).

Топологию, обычно используемую для иллюстрации SIP и RTP, часто называют SIP-трапецией (рис. 8.1). Когда Элис хочет позвонить Бобу, телефон Элис соединяется с ее прокси-сервером и прокси пытается найти Боба (часто соединяясь через его прокси). Как только соединение установлено, телефоны общаются друг с другом напрямую (если это возможно), таким образом не загружая данными ресурсы прокси-серверов.

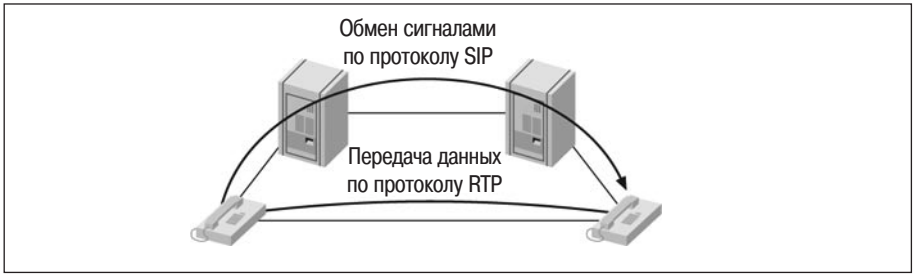



***Рис. 8.1.*** *SIP-трапеция*

SIP – не первый и не единственный используемый сегодня VoIP-протокол (среди которых можно упомянуть H. 323, MGCP, IAX и т. д.), но в настоящее время его поддерживают большинство производителей аппаратных средств. Преимущества SIP-протокола заключаются в его распространенности и гибкости архитектуры (и, как мы уже говорили, простоте!).

## Будущее

SIP заслужил свое звание протокола, который укрепил позиции VoIP. Все новые пользователи и продукты уровня предприятия должны поддерживать SIP, и любой существующий продукт сейчас не будет прода-

[1] RFC 3261, SIP: Session Initiation Protocol, стр. 9, раздел 2.

ваться, если не предлагает возможности перехода на SIP. От SIP ожидают, что предоставляемые им возможности будут намного шире, чем просто VoIP, включая возможность передавать видео, музыку и любой тип мультимедийной информации в режиме реального времени. Несмотря на то что его повсеместное использование в качестве механизма общего назначения для передачи медиа-данных пока вызывает сомнения, SIP, бесспорно, будет поставлять основную массу новых голосовых приложений в ближайшие несколько лет.

## Вопросы безопасности

Для аутентификации пользователя SIP применяет систему запрос/ответ. Исходное сообщение `INVITE` посылается на прокси-сервер, с которым желает установить соединение конечное устройство. Прокси возвращает сообщение 407 Proxy Authorization Request (запрос авторизации на прокси), содержащее случайный набор символов, который называют *временным кодом* (nonce). Этот временный код вместе с паролем используется для формирования хеша MD5, который передается в следующем сообщении `INVITE`. Если хеш MD5 соответствует хешу, сформированному прокси, клиент проходит аутентификацию.

DoS-атаки (Denial of Service – отказ в обслуживании) – наверное, самый распространенный тип атак при связи по протоколам VoIP. DoS-атакой может считаться поступление на прокси-сервер большого количества недействительных запросов `INVITE` с целью вызвать перегрузку системы. Эти атаки относительно просто реализовать, и они мгновенно отражаются на пользователях системы. SIP располагает несколькими методами минимизации эффектов от DoS-атак, но предотвратить их полностью невозможно.

SIP реализует схему, которая гарантирует, что для установления связи между вызывающим абонентом и доменом вызываемого абонента используется безопасный транспортный механизм с шифрованием (а именно Transport Layer Security, или TLS). Кроме того, защищенный запрос посылается в конечное устройство согласно локальным политикам безопасности сети. Обратите внимание, что шифрование речевых данных (то есть потока RTP) не входит в функции SIP и должно реализовываться отдельно.

Больше информации относительно вопросов безопасности в SIP, включая похищение учетных данных, подмену сервера и обрыв сеанса, можно найти в разделе 26 документации SIP RFC 3261.

## SIP и NAT

Наверное, самым большим техническим препятствием, которое должен преодолеть SIP, является проведение транзакции через сети, использующие технологию NAT. Поскольку SIP инкапсулирует адресную информацию в своих порциях данных и NAT выполняется на более низком сетевом уровне, автоматической корректировки адресной

информации не происходит. Таким образом, медиа-потоки не будут располагать правильной адресной информацией, которая необходима для завершения установления соединения при использовании NAT. Кроме того, межсетевые экраны, обычно интегрированные с NAT, не будут рассматривать поступающий медиа-поток как часть SIP-транзакции и блокируют соединение. Более новые межсетевые экраны и пограничные контроллеры сеансов (Session Border Controllers) поддерживают SIP, но это по-прежнему считается недостатком данного протокола и является источником бесконечных неприятностей для сетевых специалистов, которым приходится соединяться с конечными точками, работающими по SIP-протоколу, используя существующую сетевую инфраструктуру.

## H.323

Этот протокол, рекомендованный Международным союзом телекоммуникаций (МСТ), изначально был разработан для обеспечения транспортного механизма для видеоконференц-связи по IP-протоколу. Он стал стандартом для оборудования видеоконференц-связи, работающего по IP-протоколу, и также некоторое время пользовался популярностью как VoIP-протокол. Хотя еще ведутся жаркие дебаты по поводу того, какой из протоколов, SIP или H.323 (или IAX), будет доминировать в мире VoIP-протоколов, Asterisk почти отказалась от H.323 в пользу IAX и SIP. H.323 не завоевал особой популярности среди пользователей и компаний, хотя по-прежнему является, наверное, самым широко используемым VoIP-протоколом среди поставщиков услуг связи.

Три версии H.323, поддерживаемые в Asterisk, обрабатываются модулями chan_h323.so (поставляется с Asterisk), chan_oh323.so (доступен как бесплатное дополнение) и chan_ooh323.so (поставляется в пакете asterisk-addons).

Скорее всего, вы использовали H.323, даже не подозревая об этом: клиент NetMeeting от компании Microsoft является, наверное, наиболее популярным H.323-клиентом.

### История

H.323 был разработан МСТ в мае 1996 как средство передачи голоса, видео, данных и факсов по IP-сетям с возможностью подключения к PSTN. С тех пор H.323 пережил несколько версий и дополнений (которые вносят дополнительную функциональность в протокол), что позволяет использовать его как в простых сетях, предназначенных исключительно для VoIP, так и в сетях с более развитой топологией.

### Будущее

Будущее H.323 является предметом споров. Если брать за показатель его использование, будущее H.323 не слишком радужное, потому что

в этой связи он упоминается редко (несомненно, не так часто, как SIP). H.323 часто называют техническим соперником SIP, но, как показывают примеры других технологий, заявления такого рода редко являются решающим фактором для достижения успеха. Один из аспектов, негативно влияющих на популярность H.323, – его сложность, хотя многие отмечают, что изначально простой SIP начинает страдать тем же недугом.

H.323, безусловно, по-прежнему обслуживает основную массу мирового VoIP-трафика, но, по мере того как снижается зависимость людей от традиционных поставщиков услуг связи, будущее H.323 становится все менее предсказуемым. Хотя H.323, вероятно, и не будет предпочтительным протоколом для новых реализаций, можно с уверенностью ожидать, что нам еще некоторое время придется решать вопросы совместимости с H.323.

## Вопросы безопасности

H.323 – относительно защищенный протокол и не требует особых мер обеспечения безопасности, кроме тех, которые обычно применяются при любом обмене информацией по сети Интернет. Поскольку H.323 использует для передачи медиа-данных RTP-протокол, он не поддерживает шифрованные медиа-потоки. Использование VPN или другого шифрованного канала между конечными точками является самым распространенным способом обеспечения безопасности передачи медиа-данных. Конечно, недостатком здесь является необходимость установления этих безопасных каналов между конечными точками, что может быть не всегда удобным (или даже возможным). По мере того как VoIP все чаще используется для связи с финансовыми учреждениями, такими как банки, скорее всего, нам понадобятся расширения для наиболее популярных протоколов VoIP, обеспечивающие поддержку методов устойчивого шифрования.

## H.323 и NAT

Стандарт H.323 использует RTP-протокол IETF для переноса речевых данных между конечными точками. Поэтому в сетях с использованием NAT протоколу H.323 свойственны те же проблемы, что и SIP. Самый простой способ – переадресация соответствующих портов через NAT-устройство на внутреннего клиента.

Для получения вызовов всегда придется переадресовывать TCP-порт 1720 на клиента. Кроме того, понадобится переадресовывать UDP-порты для медиа-данных, передаваемых по протоколу RTP, и управляющих потоков RTCP (необходимый диапазон портов указан в руководстве пользователя к устройству). Более старые клиенты, такие как NetMeeting от Microsoft, также потребуют переадресации TCP-портов для туннелирования H.245 (опять же, диапазон используемых портов можно найти в руководстве пользователя).

Если за устройством NAT имеется большое количество клиентов, понадобится *шлюз*, работающий в режиме прокси. Шлюз потребует наличия интерфейса для взаимодействия закрытой IP-подсети и открытого Интернета). Тогда клиент H.323 из закрытой IP-подсети зарегистрируется на шлюзе, который будет передавать вызовы от его лица. Заметьте, что любым внешним клиентам, которые пожелают позвонить вам, тоже придется зарегистрироваться на прокси-сервере.

В настоящее время Asterisk не может выступать в роли шлюза H.323. Для этого придется использовать отдельное приложение, такое как OpenH323 Gatekeeper с открытым исходным кодом (*http://www.gnugk.org*).

## MGCP

Протокол контроля медиа-шлюзов (Media Gateway Control Protocol, MGCP) мы также получили от IETF. Хотя MGCP распространен больше, чем кому-то может показаться, он быстро сдает позиции в пользу таких протоколов, как SIP и IAX. Однако Asterisk любит протоколы, поэтому, естественно, имеет базовую поддержку и этого протокола.

MGCP описан в RFC 3435[1]. Он был разработан, чтобы максимально упростить конечные устройства (такие, как телефоны) и перенести всю логику и обработку вызовов на плечи медиа-шлюзов и агентов вызова. В отличие от SIP, MGCP использует централизованную модель. MGCP-телефоны не могут напрямую вызывать другие MGCP-телефоны; обязательно должен использоваться какой-нибудь контроллер.

Asterisk поддерживает MGCP посредством модуля chan_mgcp.so, а конечные точки определены в конфигурационном файле mgcp.conf. Поскольку Asterisk предоставляет только базовые сервисы агента вызовов, он не может эмулировать телефон MGCP (чтобы зарегистрироваться на другом MGCP-контроллере как агент пользователя, например).

Если у вас под рукой есть несколько MGCP-телефонов, их можно использовать с Asterisk. Если планируется вводить MGCP-телефоны в эксплуатацию в системе Asterisk, помните, что сообщество перешло на более популярные протоколы и поэтому необходимо соответственно планировать затраты на программную поддержку. Если это возможно (например, для телефонов Cisco), следует обновить MGCP-телефоны и перейти на протокол SIP.

## Узкоспециализированные протоколы

Наконец давайте рассмотрим два поддерживаемых в Asterisk узкоспециализированных протокола.

### Skinny/SCCP

Протокол Skinny Client Control Protocol (SCCP) является узкоспециализированным протоколом для VoIP-оборудования Cisco. Это протокол

[1] RFC 3435 пришел на смену устаревшему RFC 2705.

по умолчанию для конечных точек офисной АТС Cisco Call Manager[1]. Asterisk поддерживает Skinny, но при подключении телефонов Cisco к Asterisk, как правило, рекомендуется получить SIP-образы для всех телефонов, поддерживающих SIP, и соединяться через SIP.

### UNISTIM

Поддержка узкоспециализированного VoIP-протокола компании Nortel UNISTIM означает, что Asterisk является первой офисной АТС в истории, поддерживающей узкоспециализированные IP-терминалы двух крупнейших игроков в VoIP – Nortel и Cisco. Реализация поддержки UNISTIM – это лишь эксперимент, и система работает недостаточно устойчиво, чтобы вводить ее в эксплуатацию, но тот факт, что кто-то не поленился сделать это, демонстрирует мощь платформы Asterisk.

## Кодеки

Под *кодеками*, как правило, понимают различные математические модели, используемые для цифрового кодирования (и сжатия) аналоговой аудиоинформации. Многие из этих моделей учитывают способность человеческого мозга формировать законченное впечатление по неполной информации. Все мы видели оптические иллюзии; точно так же алгоритмы сжатия голоса используют нашу способность представлять то, что, как нам *кажется*, мы должны слышать, а не то, что мы *фактически* слышим[2]. Цель различных алгоритмов кодирования – обеспечить баланс между эффективностью и качеством[3].

Изначально под термин «*кодек*» был образован от слов КОдер/ДЕКодер – это устройство, которое выполняет преобразования между аналоговым и цифровым сигналом. Теперь этот термин, кажется, больше относится к понятиям КОмпрессия/ДЕКомпрессия.

---

[1] Cisco недавно объявила о планируемом переходе на протокол SIP в будущих продуктах.

[2] «Бриатснкие учнеые усатонвили: не вжано, как вы рсасталвятее бкувы вунрти солва, галвоне, чотб певрая и псолденяя бувкы отсавласись ниезмеынми, тгода ткест бдует вопсриинмаьтся парвиьлно. Это пориосхдит помтоу, что мы чиатем не каджую бувку в отдольенсти, а солво в цеолм». (Источник этой цитаты неизвестен, смотрите по адресу *http://www.bisso.com/ujg_archives/000228.html*.) То же самое мы делаем со звуком: если информации достаточно, наш мозг может заполнять пробелы.

[3] Для аудио-CD качество намного важнее экономии полосы пропускания, поэтому квантование звука выполняется с разрядностью 16 бит (умноженной на 2, поскольку это стерео), с частотой дискретизации 44 100 Гц. Учитывая то, что CD был изобретен в конце 1970-х годов, это была довольно впечатляющая нагрузка в то время. Телефонная сеть не требует такого уровня качества (и нуждается в оптимизации полосы пропускания), поэтому телефонные сигналы кодируются с использованием 8 бит и частотой дискретизации 8000 Гц.

Прежде чем перейти к каждому кодеку в отдельности, рассмотрим табл. 8.1, где представлены краткие данные, которые можно использовать для справки.

*Таблица 8.1. Краткие справочные данные для кодеков*

| Кодек | Скорость передачи данных, Кбит/с | Необходимость лицензии |
|---|---|---|
| G.711 | 64 | Не нужна |
| G.726 | 16, 24, 32 или 40 | Не нужна |
| G.729A | 8 | Нужна (не нужна для транзитной пересылки) |
| GSM | 13 | Не нужна |
| iLBC | 13,3 (кадры по 30 мс) или 15,2 (кадры по 20 мс) | Не нужна |
| Speex | Переменная (между 2,15 и 22,4) | Не нужна |

## G.711

G.711 – основной кодек PSTN. В сущности, при упоминании ИКМ (обсуждается в предыдущей главе) в связи с телефонной сетью можно смело иметь в виду G.711. Используется два метода компандирования: μlaw в Северной Америке и alaw во всем остальном мире. Любой из них обеспечивает передачу 8-битового слова 8000 раз в секунду. Если произвести вычисления, можно увидеть, что это потребует передачи 64 000 бит/с.

Многие скажут вам, что G.711 – это кодек без сжатия. Это не вполне так, поскольку компандирование считается формой сжатия. На самом деле G.711 является базовым кодеком, от которого были произведены все остальные.

G.711 налагает минимальную (практически нулевую) нагрузку на ЦП.

## G.726

Этот кодек активно использовался некоторое время (его называли G.721, но сейчас он вышел из употребления) и является одним из исходных кодеков со сжатием. Эта технология известна как адаптивная дифференциальная импульсно-кодовая модуляция (Adaptive Differential Pulse-Code Modulation, ADPCM), она обеспечивает разную скорость передачи данных. Чаще всего используются скорости 16, 24 и 32 Кбит/с. На момент написания данной книги Asterisk поддерживала только ADPCM-32, несомненно, самую популярную скорость передачи данных для этого кодека.

G.726 предлагает качество практически такое же, как у G.711, но использует только половину полосы пропускания. Это возможно потому, что он отправляет не результат измерения, а только достаточную информацию для описания разницы между текущим и предыдущим за-

мерами. G.726 потерял популярность в 1990-х годах из-за неспособности передавать сигналы модема и факсов, но сейчас она вновь возвращается благодаря обеспечиваемому им соотношению пропускная способность/нагрузка на ЦП. G.726 особенно привлекателен, потому что не требует от системы проведения большого объема вычислений.

## G.729A

Учитывая, насколько малую полосу пропускания использует кодек G.729A, он обеспечивает впечатляющее качество звука. Делает он это за счет технологии Conjugate-Structure Algebraic-Code-Excited Linear Prediction (CS-ACELP)[1]. G729A является запатентованным продуктом, поэтому его нельзя использовать без лицензии; однако он чрезвычайно популярен и, соответственно, поддерживается многими разными телефонами и системами.

Чтобы достичь такой значительной степени сжатия, этот кодек требует такой же значительной работы от ЦП. В системе Asterisk использование кодеков с большой степенью сжатия быстро приводит к перегрузке ЦП.

Для G.729A требуется пропускная способность 8 Кбит/с.

## GSM

GSM – самый любимый кодек Asterisk. Он не обременен лицензионными соглашениями, как G.729A, и предлагает превосходную производительность, если учитывать требования, которые он предъявляет к ЦП. Качество получаемого звука, в общем, считается ниже, чем обеспечивает G.729A, но это преимущественно субъективное мнение; обязательно попробуйте его. Скорость передачи данных GSM – 13 Кбит/с.

## iLBC

Internet Low Bitrate Codec (iLBC)[2] обеспечивает привлекательное сочетание низкого коэффициента использования полосы пропускания

---

[1] CELP – популярный метод сжатия речи. Моделируя математически различные способы воспроизведения звуков человеком, можно построить книгу кодов. Вместо того чтобы посылать реальный дискретный звук, определяется соответствующий ему код. Кодеки CELP берут эту информацию (которая сама по себе будет создавать совершенно механический звук) и пытаются вернуть ей индивидуальные особенности. (Конечно, делается намного большее.) На странице Джейсона Вудворда (Jason Woodward) Speech Coding (Кодирование речи) (*http://www-mobile.ecs.soton.ac.uk/speech_codecs/*) можно найти полезную информацию для тех, кто не хочет вдаваться в математические подробности. Тем не менее материал довольно тяжелый, придется напрячь извилины.

[2] Кодек для низких скоростей передачи данных в Интернете. – *Примеч. науч. ред.*

и приемлемого качества. Он особенно хорошо подходит для обеспечения целесообразного качества в сетевых соединениях с потерями.

Естественно, Asterisk поддерживает его (и поддержка этого кодека другими системами тоже растет), но он не так популярен, как кодеки ITU, и, таким образом, может не поддерживаться обычными IP-телефонами и коммерческими VoIP-системами. IETF RFC 3951 и 3952 опубликованы в поддержку iLBC, и iLBC находится на пути к стандартизации IETF.

iLBC использует сложные алгоритмы для достижения таких высоких уровней сжатия, поэтому довольно сильно загружает ЦП при использовании в Asterisk.

iLBC можно использовать без всяких авторских отчислений, но владелец патента на iLBC, Global IP Sound (GIPS), желает получать информацию обо всех случаях его применения в коммерческих целях. Для этого надо скачать и распечатать копию лицензии на использование iLBC, подписать ее и отправить GIPS. Почитать о iLBC и лицензии на его использование можно по адресу *http://www.ilbcfreeware.org*.

iLBC используется для каналов со скоростью передачи данных 13,3 Кбит/с (кадры по 30 мс) и 15,2 Кбит/с (кадры по 20 мс).

## Speex

Speex – это кодек с переменной скоростью передачи цифровых данных (Variable Bitrate, VBR). Это означает, что он может динамически менять скорость передачи данных в ответ на изменение условий сети. Он предлагается как в узкополосном, так и в широкополосном вариантах в зависимости от того, какого качества звук требуется получить (телефонного или лучше).

Speex – абсолютно бесплатный кодек, лицензированный по версии Xiph.org лицензии BSD.

В Интернете доступен проект спецификации Speex. Больше информации о Speex можно найти на его странице (*http://www.speex.org*).

Speex может использоваться для каналов со скоростью передачи данных от 2,15 до 22,4 Кбит/с благодаря его способности менять свою скорость передачи данных.

## MP3

Конечно же, MP3 – это кодек. Вообще говоря, он называется Moving Picture Experts Group Audio Layer 3 Encoding Standard[1]. С таким именем неудивительно, что его называют MP3! В Asterisk кодек MP3 обычно используется для музыки во время ожидания (Music on Hold, MoH).

[1] Если хотите почитать о звукозаписи в формате MPEG, найдите в Сети статью Дэвиса Пэна (Davis Pan) под названием «A Tutorial on MPEG/Audio Compression».

MP3 не предназначен для передачи речи по телефонным сетям, поскольку он оптимизирован для музыки, а не голоса. Тем не менее он очень популярен в системах телефонной связи VoIP для воспроизведения музыки во время ожидания.

Не забывайте, что обычно для трансляции музыки необходима лицензия. Многие полагают, что вполне законно могут использовать радиостанцию или CD как источник музыки во время ожидания, но в большинстве случаев это не так.

# Качество и класс предоставляемых услуг передачи данных

Качество обслуживания, или, как чаще всего говорят, *QoS* (Quality of Service), – характеристика, определяющая качество и класс услуг по передаче потока данных, предоставляемой пользователю сетью и являющейся критичной по времени. Жестких норм здесь не существует, но обычно считается, что нормальное плавное течение беседы возможно, если звук, производимый говорящим, доставляется к уху слушателя в течение 150 мс. Если задержка превышает 300 мс, участники беседы начнут перебивать друг друга. При задержке выше 500 мс нормальный разговор невозможен.

Кроме выполнения требований по времени, необходимо гарантировать, что передаваемая информация приходит неповрежденной. Потеря слишком большого количества пакетов обусловит невозможность полного восстановления дискретизированного аудиосигнала на дальнем конце, и пробелы в данных будут слышаться как помехи или, в более тяжелых случаях, пропуски целых слов или предложений. Даже утрата 5% пакетов может сильно повредить сети VoIP.

## TCP, UDP и SCTP

Для передачи данных по сети, работающей по IP-протоколу, используется один из трех обсуждаемых здесь транспортных протоколов.

### Transmission Control Protocol

Transmission Control Protocol (TCP) практически не используется для VoIP, поскольку, хотя у него имеются механизмы обеспечения гарантированной доставки, они фактически не используются. TCP склонен создавать больше проблем, чем решать их, в большинстве соединений и может применяться только в соединениях между двумя конечными точками с чрезвычайно малым временем ожидания.

Назначение TCP – гарантировать доставку пакетов. Для этого реализуется несколько механизмов, таких как нумерация пакетов (для восстановления блоков данных), подтверждение доставки и повторный за-

прос утерянных пакетов. В мире VoIP быстрая доставка пакетов в конечную точку является первостепенной задачей, но за 20 лет использования сотовой связи мы научились спокойно относиться к недостатку нескольких пакетов[1].

Тщательная обработка, управление состоянием и подтверждение доставки – все эти характеристики делают TCP прекрасным протоколом для передачи больших объемов данных, но он просто недостаточно эффективен для передачи медиа-данных в реальном масштабе времени.

## User Datagram Protocol

В отличие от TCP, User Datagram Protocol (UDP) не предлагает никаких гарантий доставки. Пакеты передаются по проводам так быстро, как это возможно, и выпускаются в мир без всякой информации о том, достигли они пункта своего конечного назначения или нет. Поскольку UDP не дает никаких гарантий доставки данных[2], его эффективность обеспечивается очень небольшими затратами на транспортировку.

TCP является более «сознательным» протоколом, потому что полоса пропускания распределяется между клиентами, соединяющимися с сервером, более равномерно. С увеличением UDP-трафика возможна перегрузка сети.

## Stream Control Transmission Protocol

Одобренный IETF в качестве предлагаемого стандарта в RFC 2960, SCTP является относительно новым транспортным протоколом. С самого начала он разрабатывался как протокол, лишенный недостатков TCP и UDP и предназначенный в первую очередь для сервисов, обычно предоставляемых коммутируемыми телефонными сетями.

Некоторыми из целей SCTP были:

- Лучшие техники предотвращения перегрузок (в частности, предотвращение атак типа «отказ в обслуживании»).
- Строго упорядоченная доставка данных.
- Более низкая задержка для улучшения передачи в режиме реального времени.

Разработчики SCTP надеялись создать надежный протокол для передачи SS7 и других типов сигналов PSTN по IP-сети, избавленный от основных недостатков TCP и UDP.

---

[1] В телефонной связи важен порядок поступления пакетов, потому что звук обрабатывается и отправляется вызывающей стороне так быстро, насколько это возможно. Однако при наличии буфера колебаний задержки порядок поступления уже становится не так критичен, поскольку в этом случае обеспечивается небольшое временное окно, в течение которого может быть изменен порядок пакетов перед передачей вызывающей стороне.

[2] Помните, что протоколы или приложения верхнего уровня могут реализовывать собственные системы подтверждения доставки пакетов.

## Дифференцированное обслуживание

Дифференцированное обслуживание, или DiffServ, – не столько механизм QoS, сколько метод, с помощью которого можно маркировать трафик и обеспечивать ему специальное обслуживание. Очевидно, что DiffServ может способствовать обеспечению QoS, предоставляя преимущество определенным типам пакетов над другими.

Хотя, несомненно, это повышает шансы VoIP-пакета быстро пройти через все соединения, но не дает твердых гарантий.

## Гарантированное обслуживание

Гарантированное качество и класс предоставляемых услуг обеспечивает PSTN. Для каждого разговора используется выделенный только под этот звонок канал со скоростью передачи данных 64 Кбит/с; пропускная способность гарантируется. Точно так же протоколы, предлагающие гарантированное обслуживание, могут обеспечить выделение под обслуживаемое соединение необходимой полосы пропускания. Как для любой сетевой технологии с пакетированием, эти механизмы, как правило, лучше всего работают в условиях, когда объем трафика ниже максимально допустимых уровней. Если соединение достигает своих предельных значений, практически невозможно избежать ухудшения качества обслуживания.

### MPLS

Multiprotocol Label Switching (MPLS) – это метод разработки и управления моделями сетевого трафика независимо от таблиц маршрутизации третьего (сетевого) уровня. Суть работы протокола заключается в присваивании сетевым пакетам коротких меток (кадров MPLS), которые затем используются маршрутизатором для пересылки пакетов на выходной маршрутизатор MPLS и в итоге – к их окончательному месту назначения. Традиционно маршрутизаторы принимают независимое решение о пересылке на основании поиска в IP-таблице при каждом переходе в сети. В сети MPLS такой поиск выполняется только один раз, когда пакет входит в MPLS-облако на входном маршрутизаторе. После этого пакету назначается поток, называемый Label Switched Path (LSP) и идентифицируемый по метке. Метка используется как индекс поиска в таблице пересылки MPLS, и пакет проходит по LSP независимо от решений маршрутизации третьего уровня. Это позволяет администраторам больших сетей тонко настраивать решения по маршрутизации и использовать сетевые ресурсы с максимальной эффективностью. Кроме того, с меткой может быть связана информация, определяющая приоритетность пакета при пересылке.

### RSVP

В MPLS нет метода для динамического установления LSP, но для этого в сочетании с MPLS можно использовать Reservation Protocol (RSVP).

RSVP – это протокол обмена сигналами, используемый для упрощения задач по установлению LSP и передачи информации о возникающих проблемах на входной маршрутизатор MPLS. Преимущество использования RSVP в сочетании с MPLS – сокращение затрат на администрирование. Если не использовать RSVP с MPLS, придется вручную конфигурировать все метки и каждый путь на всех маршрутизаторах. Применение RSVP делает сеть более динамичной за счет передачи функции управления метками маршрутизаторам. Таким образом, сеть становится более чутко реагирующей на изменяющиеся условия и может быть настроена на изменение путей исходя из определенных условий, например если какой-то из путей недоступен (возможно, по причине выхода из строя маршрутизатора). В этом случае конфигурация маршрутизатора сможет использовать RSVP для распределения новых меток среди маршрутизаторов MPLS-сети без всякого вмешательства человека (или при минимальном вмешательстве).

## Негарантированное обслуживание

Самый простой и дешевый подход к QoS – не предоставлять качества услуг вообще. Это называется негарантированным обслуживанием. Вероятно, звучит не очень хорошо, но этот метод может очень неплохо работать. Любой вызов VoIP, проходящий по открытой сети Интернет, практически наверняка будет вызовом с негарантированным обслуживанием, поскольку механизмы QoS в этой среде еще не получили широкого распространения.

# Эхо

Возможно, вы не осознаете этого, но проблема эха существует в PSTN так же долго, как существуют телефоны. Вероятно, вы не часто сталкивались с ней, потому что телекоммуникационная отрасль потратила огромные суммы денег на разработку дорогих эхоподавляющих устройств. Также, если конечные точки физически располагаются на небольшом расстоянии, например когда вы звоните своему соседу, живущему на одной с вами улице, задержка минимальна и все сигналы возвращаются настолько быстро, что полностью имитируют местный эффект[1], обычно создаваемый телефоном. То есть суть в том, что при местных звонках эхо присутствует в большинстве случаев, но абонент не может различить его в обычном телефоне, потому что оно возвращается практически мгновенно. Чтобы понять это, представьте следующее: когда вы находитесь в комнате, все сказанное вами отражается от стен и потолка (и, вероятно, пола, если нет ковра) и возвращается к вам, но это не создает никаких проблем, потому что происходит настолько быстро, что вы не улавливаете задержки.

[1] Как говорилось в главе 7, местный эффект – это функция телефона, которая обеспечивает возвращение части звукового сигнала в ухо говорящего, чтобы создать эффект более естественного разговора.

Причина, по которой в телефонной системе VoIP, такой как Asterisk, может появиться эхо, в том, что введение VoIP-телефона приводит к возникновению небольшой задержки. Прохождение пакетов от телефона на сервер (и обратно) занимает несколько миллисекунд. Если вдруг возникает ощутимая задержка, вы можете слышать эхо, которое было там всегда, но никогда не приходило с задержкой.

## Почему возникает эхо

Прежде чем приступить к обсуждению мер по борьбе с эхом, давайте рассмотрим, почему эхо возникает в аналоговом мире.

Если вы слышите эхо, проблема не в телефоне, а в дальнем конце линии. И наоборот, эхо, слышимое на дальнем конце, формируется на вашей стороне. Эхо может быть обусловлено тем, что местной аналоговой линии связи приходится передавать и получать сигналы по одной и той же паре проводов. Если эта линия не сбалансирована по электрическим параметрам или если к ней подключен телефон низкого качества, получаемые ею сигналы могут отражаться назад в сеть, становясь частью возвращаемых данных. Когда этот отраженный сигнал возвратится к вам, вы услышите слова, которые произнесли несколько мгновений назад. Люди будут различать эхо при задержке, превышающей определенную величину (возможно, от 20 мс для некоторых). При увеличении задержки эхо начнет раздражать.

В дешевом телефоне эхо может формироваться в теле трубки. Вот почему некоторые дешевые IP-телефоны могут создавать эхо, даже если в соединении нет ни одной аналоговой линии[1]. В мире VoIP эхо обычно обусловлено или присутствием аналоговой линии где-то в соединении, или применением дешевого терминала, отражающего часть сигнала (например, обратная связь через устройство hands-free или плохую трубку либо гарнитуру). Чем выше задержка в сети, тем более надоедливым может быть это эхо.

## Устранение эха в каналах Zaptel

В конфигурационном файле zconfig.h можно выбрать один из ряда предлагаемых алгоритмов эхоподавления. По умолчанию используется MARK2. Поэкспериментируйте с различными эхокомпенсаторами, чтобы выбрать тот, который лучше всего подходит для вашей среды. Также Asterisk предлагает опцию в файле zconfig.h, которая позволяет сделать эхоподавление более агрессивным. Ее можно активировать, раскомментировав следующую строку:

```
#define AGGRESSIVE_SUPPRESSOR
```

[1] На самом деле трубка любого телефона, будь он традиционным или VoIP, является аналоговой линией.

Заметьте, что агрессивное эхоподавление может создать эффект портативной полудуплексной радиостанции. Оно должно быть активировано, только если все остальные методы снижения эха не обеспечили результата.

Активация эхоподавления для интерфейсов Zaptel осуществляется в файле zapata.conf. В стандартной конфигурации эхоподавление активируется строкой `echocancel=yes`. `echocancelwhenbridged=yes` обеспечит эхоподавление для звонков, проходящих через TDM. Хотя такие звонки не должны требовать эхоподавления, это может улучшить их качество.

Если эхоподавление активировано, эхокомпенсатор распознает эхо в линии во время звонка. Поэтому эхо может быть слышимо в начале разговора и со временем уменьшаться. Чтобы избежать этого, можно прибегнуть к методу, называемому *тренировкой эхоподавления*, при котором в линии в начале звонка на мгновение отключается звук и передается тональный сигнал, по которому может быть определена величина эха. Это позволяет Asterisk быстрее устранять эхо. Тренировка эхоподавления активируется строкой `echotraining=yes`.

### Аппаратное эхоподавление

Программное эхоподавление – не самый эффективный способ борьбы с эхом. Если планируется развертывание системы хорошего качества, потратьте дополнительные средства на платы, снабженные устройствами аппаратного эхоподавления. Такие платы несколько дороже обычных, но они быстро окупятся, поскольку обеспечат снижение нагрузки на ЦП и сберегут ваши нервы благодаря сокращению жалоб пользователей.

## Asterisk и VoIP

Для вас не должно быть сюрпризом, что Asterisk любит работать с VoIP. Но для этого ей надо знать, какую функцию выполнять: клиента, сервера или и того и другого. Одна из наиболее сложных и часто сбивающих с толку концепций в Asterisk – схема присваивания имен при аутентификации входящих и исходящих вызовов.

### Пользователи, и равноправные участники, и друзья – о, боже!

Соединения, устанавливаемые с нами или нами, определены в файлах iax.conf и sip.conf как `user` (пользователь) и `peer` (равноправный участник). Соединения, которые могут выполняться в обоих направлениях, могут быть определены как `friend` (друг). При определении, в каком направлении происходит аутентификация, всегда важно посмотреть на направление каналов с точки зрения Asterisk, поскольку соединения принимаются и создаются сервером Asterisk.

## Соединения user

Соединение, определенное как `user`, – это любая система/пользователь/конечная точка, которой мы разрешаем соединяться с нами. Помните, что описание `user` не обеспечивает метода вызова этого пользователя; тип `user` используется просто для создания канала для входящих звонков[1]. В описании `user` потребуется задать имя контекста для обозначения места диалплана (в файле extensions.conf), где будет начинаться обработка аутентифицированных звонков.

## Соединения peer

Соединение типа `peer` является исходящим. Представим это так: пользователи (users) звонят нам, тогда как мы звоним равноправным участникам (peers). Поскольку равноправные участники не звонят нам, описание `peer` обычно не требует задания имени контекста. Однако есть одно исключение: если звонки, берущие начало в вашей системе, возвращаются в вашу же систему, входящие звонки (которые берут начало на SIP-прокси, а не на агенте пользователя) будут сопоставляться с описанием `peer`. Контекст `default` должен обрабатывать эти входящие звонки соответствующим образом, хотя предпочтительнее, чтобы контексты были определены для каждого `peer` отдельно[2].

Чтобы знать, куда отправлять вызов, необходимо иметь информацию о местонахождении хоста в Интернете (то есть знать его IP-адрес). Местоположение `peer` может быть определено или статически, или динамически. Динамический `peer` конфигурируется с помощью строки `host=dynamic`, размещаемой под заголовком описания. Поскольку IP-адрес динамического `peer` может меняться постоянно, он должен регистрироваться на сервере Asterisk, чтобы его IP-адрес был известен и звонки могли успешно направляться к нему. Если удаленным концом является другой сервер Asterisk, необходимо использовать выражение `register`, что обсуждается ниже.

## Соединения friend

Определение типа `friend` является сокращенной записью для соединения, которое может быть и `user`, и `peer`. Однако соединения, являющиеся и `user`, и `peer`, не всегда определяются так, потому что индивидуаль-

---

[1] В SIP это не всегда так. Если конечная точка является прокси-сервисом SIP (в противоположность агенту пользователя), Asterisk будет выполнять аутентификацию на основании описания `peer`, сравнивая IP-адрес и порт в поле `Contact` SIP-заголовка с именем хоста (и портом, если он задан), определенным для этого равноправного участника (если порт не задан, будет использоваться тот, который определен в разделе `[general]`). Подробнее опция SIP `insecure` обсуждается в приложении A.

[2] Больше информации по этому вопросу можно найти в обсуждении SIP-опции `context` в приложении A.

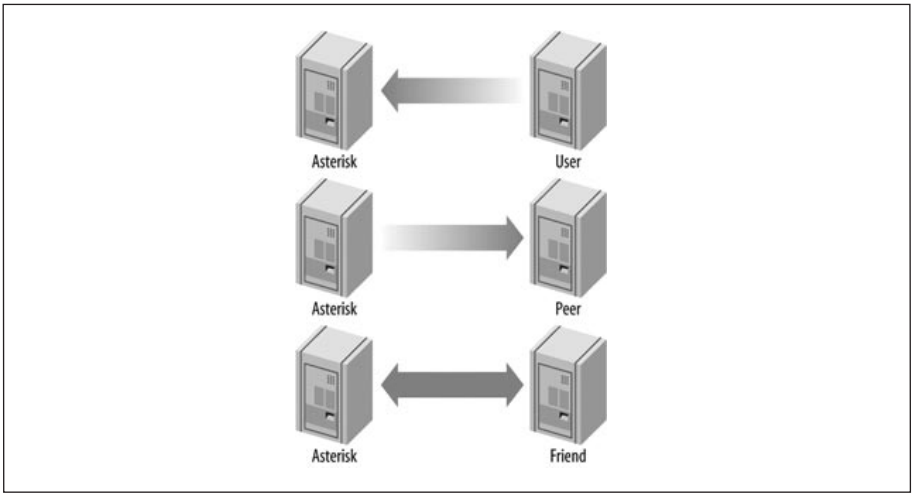



***Рис. 8.2.*** *Источник вызова относительно Asterisk для соединений типа user, peer и friend*

ное описание каждого направления создания вызова (использование двух описаний, `user` и `peer`) обеспечивает возможность более тонкой настройки и управления каждым отдельно взятым соединением. На рис. 8.2 показан поток управления аутентификацией по отношению к Asterisk.

## Выражения register

Выражение `register` – это средство сообщить удаленному равноправному участнику сети, где в Интернете находится ваш сервер Asterisk. Asterisk использует выражения `register` для аутентификации у удаленных поставщиков сервисов, если вы используете динамические IP-адреса или если ваш IP-адрес не зарегистрирован у поставщика. Возможны ситуации, когда выражение `register` не требуется, но, чтобы продемонстрировать случаи, когда выражение `register` необходимо, рассмотрим следующий пример.

Допустим, имеется удаленный равноправный участник сети, предоставляющий вам сервисы DID. Когда кто-то вызывает номер +1-800-555-1212, звонок поступает по физической сети PSTN к вашему поставщику сервисов и на его сервер Asterisk, возможно, через T1-линию. После этого данный вызов направляется по Интернету на ваш сервер Asterisk.

Ваш поставщик услуг будет располагать описанием вашего сервера Asterisk в одном из конфигурационных файлов, sip.conf или iax.conf (в зависимости от того, выполняется ли соединение по протоколу SIP или IAX соответственно). Если вы получаете вызовы только от этого поставщика сервисов, вы определили бы их тип как `user` (если бы это

была другая система Asterisk, вы могли бы быть определены в ней как `peer`).

Теперь, допустим, ваш сервер использует ваше домашнее соединение с Интернетом с динамическим IP-адресом. Поставщик услуг имеет статический IP-адрес (или, возможно, полностью определенное доменное имя), которое указано в вашем конфигурационном файле. Поскольку у вас динамический адрес, поставщик сервисов в своем конфигурационном файле указывает `host=dynamic`. Чтобы знать, куда направлять ваш звонок на номер +1-800-555-1212, поставщику сервисов необходимо знать ваше местонахождение в Интернете. Вот где понадобится выражение `register`.

Выражение `register` – это средство аутентификации и сообщения `peer` своего местонахождения. В разделе `[general]` своего конфигурационного файла поместите выражение, аналогичное данному:

```
register => имяпользователя:секрет@мой_удаленнный_равноправный_участник
```

Убедиться в успешности регистрации можно с помощью команд `iax2 show registry` и `sip show registry` из консоли Asterisk.

# Безопасность VoIP

В данной книге мы можем лишь коснуться сложного и широкого вопроса безопасности VoIP; поэтому, прежде чем углубиться в него, мы хотели бы направить вас к VoIP Security Alliance (*http://www.voipsa.org*). Этот фантастический ресурс имеет превосходную рассылку, техническую документацию, практические рекомендации и общий перечень всех материалов, касающихся безопасности VoIP. Как и сообщения электронной почты, речевые данные тоже могут быть подвергнуты атакам корыстного или криминального характера. Хорошие парни на VoIPSA делают все, что в их силах, чтобы мы могли справиться с этими проблемами сейчас, до того как они станут эпидемией. Из книг, посвященных этому вопросу, мы рекомендуем самую лучшую – «Hacking Exposed VoIP» (издательство McGraw-Hill Osborne Media) Дэвида Эндлера (David Endler) и Марка Коллиера (Mark Collier). Те, кто отвечает за развертывание любой системы VoIP, должны знать этот материал.

## Спам по сети интернет-телефонии (СПИТ)

Нам не хочется думать об этом, но мы знаем, что он будет. Чтобы предсказать это, достаточно того простого факта, что в этом мире есть люди, в которых отсутствие определенных социальных навыков сочетается с тупой жадностью, и такие парни думают только о том, как наводнить Интернет огромной массой электронной почты. Эти же ребята, недолго думая, начнут делать то же самое с голосовой связью. Мы уже знаем, что значит утопать в звонках систем продаж по телефону; а теперь попытайтесь представить ситуацию, когда рассылка голосового спама не

стоит телемаркетинговой фирме практически ни копейки. Никакие меры не остановили спам по электронной почте и, вероятно, не остановят голосовой спам, поэтому борьба с ним будет нашей задачей.

## Шифрование звука с помощью безопасного RTP

Если можно перехватывать пакеты, исходящие из системы Asterisk, значит, можно извлекать аудиоданные из RTP-потоков. Эти данные могут поставляться в автономном режиме в систему обработки речи, которая будет слушать ключевые слова, такие как «номер кредитной карты» или «пин», и предоставлять эти данные тому, кто заинтересован в них. Поток также может быть проанализирован на предмет встроенных DTMF-тонов, что представляет опасность, потому что многие сервисы запрашивают ввод пароля и информации кредитной карты через номеронабиратель. Также и в деловой сфере, имея возможность сбора и анализа аудиоданных, можно выведать стратегически важную информацию.

Использование безопасного RTP (Secure RTP, SRTP) может помочь справиться с этой проблемой за счет шифрования RTP-потоков; но Asterisk не поддерживала SRTP на момент написания данной книги. Работы по обеспечению поддержки SRTP ведутся (для готовящейся к выпуску версии есть патч, но на данный момент неизвестно, будет ли он перенесен в более раннюю версию 1.4).

## Спуфинг

В традиционной телефонной сети очень сложно действовать от лица кого-либо. Ваша деятельность может быть (и будет) отслежена, и полномочные органы быстро положат конец забавам. В мире IP намного проще сохранять анонимность. Поэтому несложно догадаться, что орды предприимчивых злоумышленников станут охотно звонить в компанию по выдаче кредитных карт или в банк от вашего имени. Если не будет придуман надежный механизм борьбы со спуфинг-мошенничеством[1], мы быстро поймем, что не можем доверять звонкам по VoIP.

## Что можно сделать

Первое, что надо помнить при рассмотрении вопросов безопасности в VoIP-системе, – VoIP основывается на сетевых протоколах, то есть анализ необходимо проводить с данной точки зрения. Мы не хотим

[1] Спуфинг – мошенничество с использованием реквизитов солидных компаний, направленное на получение конфиденциальной информации с целью хищения денег. Как правило, с помощью компьютерных технологий имитируется реально существующий банковский сайт или он размещается на поддельном сайте; таким образом, вводимые обманутыми пользователями данные поступают на поддельный сайт. – *Примеч. науч. ред.*

сказать, что должны игнорироваться традиционные меры защиты, используемые в телекоммуникациях, но необходимо обратить внимание на сеть, лежащую в основе.

## Базовая безопасность сети

Самое эффективное, что можно сделать, – обеспечить защищенный доступ к сети телефонной связи. Применение межсетевых экранов и виртуальных локальных сетей (VLAN) – примеры того, как это можно реализовать. По умолчанию сеть телефонной связи должна быть доступной только для того, в чем есть необходимость. Например, если программные телефоны не используются, клиентские ПК не должны иметь доступа к сети телефонной связи.

**Разделение речевого трафика и трафика данных.** Если нет необходимости в передаче речи и данных по одной сети, возможно, будет выгодно разделить их (это может иметь также и другие преимущества, такие как упрощение конфигурации QoS). Не является чем-то из ряда вон выходящим полная реализация сети телефонной связи в абсолютно изолированной локальной сети, использующей существующую кабельную разводку CAT3 и заканчивающейся недорогими сетевыми коммутаторами. Это также может быть дешевле.

**Демилитаризованная зона (DMZ).** Размещение системы VoIP в демилитаризованной зоне может обеспечить дополнительный уровень защиты локальной сети, предоставляя при этом возможность соединения для соответствующих приложений. Если система VoIP будет подвергнута несанкционированному доступу, будет намного сложнее использовать ее для распространения атаки на остальную сеть, поскольку она не является доверенной. Но даже если развертывание системы выполняется в демилитаризованной зоне, любой необычный трафик, исходящий из системы, должен находиться под подозрением.

**Укрепление сервера.** Укрепление сервера Asterisk является критически важным. Это обеспечивает не только преимущества по производительности (выполнение второстепенных процессов может съедать ценные ресурсы ЦП и оперативной памяти); устранение всего ненужного сократит шанс того, что уязвимость операционной системы может быть использована для получения доступа и организации атак на другие части вашей сети.

Запуск Asterisk под учетной записью, не принадлежащей администратору, – важнейшая составляющая укрепления системы. Подробнее этот вопрос рассматривается в главе 11.

## Шифрование

Даже несмотря на то что Asterisk до сих пор не обеспечивает полной поддержки SRTP, трафик VoIP можно шифровать. Например, между узлами связи может использоваться VPN. В этом случае необходимо исходить из затрат на обеспечение необходимой производительности,

но, как правило, это очень эффективный и относительно простой в реализации способ защиты трафика VoIP.

### Физическая безопасность

Нельзя пренебрегать и физической безопасностью. Все оконечное оборудование (такое, как коммутаторы, маршрутизаторы и сама офисная АТС) должно размещаться в безопасном месте с возможностью доступа к нему только людей, имеющих на то специальное разрешение. На стороне пользователя (например, под столом) довольно сложно обеспечить физическую безопасность, но, если сеть отвечает только знакомым устройствам (например, если в DHCP все выдаваемые IP-адреса жестко привязаны к, MAC-адресам устройств, которые известны), риск неавторизованного вторжения можно несколько снизить.

## Заключение

Если верить слухам, распространяемым в телекоммуникационной отрасли, можно подумать, что VoIP – это будущее телефонии. Но для Asterisk VoIP из области «это мы уже проходили». Для Asterisk будущее телефонии намного более захватывающее. Но мы обратимся к этому несколько позднее, в главе 15. В следующей главе мы собираемся углубиться в одну более революционную и мощную концепцию Asterisk: AGI (Asterisk Gateway Interface) – шлюзовой интерфейс Asterisk.

# 9

# Шлюзовой интерфейс Asterisk (AGI)

*Даже он, кому большинство из того, что другим людям кажется достаточно разумным, представляется довольно бестолковыми, полагал, что это было достаточно разумным.*

– Дуглас Адамс «Лосось сомнений»

Шлюзовой интерфейс Asterisk, или AGI, предоставляет стандартный интерфейс, посредством которого внешние программы могут управлять диалпланом Asterisk. Как правило, сценарии AGI используются для реализации расширенной логики, соединения с реляционными базами данных (такими, как PostgreSQL или MySQL) и доступа к другим внешним ресурсам. Передача управления диалпланом внешнему сценарию AGI позволяет Asterisk без труда реализовывать задачи, выполнение которых в противном случае было бы сложным или невозможным.

В данной главе рассматриваются основы использования AGI. Она не научит вас программировать, напротив, здесь предполагается, что читатель уже является достаточно квалифицированным разработчиком, чтобы понимать, как создаются AGI-программы. Если вы не знаете, как писать компьютерные программы, вероятно, эта глава не для вас. Пропустите ее и переходите к следующей.

По ходу данной главы будет написана AGI-программа с использованием разных языков программирования: Perl, PHP и Python. Однако обратите внимание, что, поскольку Asterisk предоставляет стандартный интерфейс для AGI-сценариев, эти сценарии могут быть написаны практически на любом современном языке программирования. Мы решили остановиться на Perl, PHP и Python, потому что эти языки чаще всего используются для программирования AGI.

# Основы обмена информацией с AGI

AGI не реализует API для программирования. AGI-сценарии взаимодействуют с Asterisk по каналам связи (посредством *описателей файла*, выражаясь на языке программистов), которые известны как STDIN, STDOUT и STDERR. Большинство программистов знают эти каналы, но все-таки, на случай если вы не знакомы с ними, они будут рассмотрены здесь.

## Что такое STDIN, STDOUT и STDERR

STDIN, STDOUT и STDERR – это каналы, по которым программы в UNIX-подобных средах обмениваются информацией с внешними программами. STDIN, или *стандартный ввод*, – это информация, передаваемая в программу или с клавиатуры, или из другой программы. В нашем случае данные, поступающие от самой Asterisk, приходят в описатель файла STDIN. STDOUT, или *стандартный вывод*, – это описатель файла, применяемый сценарием AGI для передачи информации в Asterisk. И наконец, сценарий AGI может использовать описатель файла STDERR (*стандартная ошибка*) для записи сообщений об ошибке в консоль Asterisk.

Подытожим эти три концепции обмена информацией:

- Сценарий AGI читает из STDIN для получения информации от Asterisk.
- Сценарий AGI записывает данные в STDOUT для отправки информации в Asterisk.
- Сценарий AGI может записывать данные в STDERR для отправки отладочной информации в консоль Asterisk.

На данный момент запись в STDERR из сценария AGI обеспечивает запись данных только в первую консоль Asterisk – точнее, в первую консоль Asterisk, запущенную с параметром -c.

Это довольно неудачно, и мы надеемся, что данный недостаток будет вскоре устранен разработчиками Asterisk.

Если для запуска Asterisk используется программа safe_asterisk (скорее всего, вы так и поступаете), она запускает удаленную консоль на TTY9. (Проверьте, получите ли вы интерфейс командной строки Asterisk, нажав сочетание клавиш Ctrl+Alt+F9.) Это означает, что вся отладочная информация AGI будет выводиться только в той удаленной консоли. Вероятно, вы захотите деактивировать эту консоль в safe_asterisk, чтобы получить возможность видеть отладочную информацию в другой консоли. (Также, вероятно, эту консоль необходимо отключить из соображений безопасности: вам не нужно, чтобы каждый, подошедший к вашему серверу Asterisk, имел доступ к консоли без всякой аутентификации.)

## Стандартная схема обмена информацией с AGI

Обмен информацией между Asterisk и сценарием AGI идет по заранее установленной схеме. Перечислим все этапы и затем последовательно рассмотрим один из примеров сценария AGI, поставляемого с Asterisk.

Когда сценарий AGI запускается, Asterisk передает в него список переменных и их значений. Эти переменные могут выглядеть примерно так:

```
agi_request: test.py
agi_channel: Zap/1-1
agi_language: en
agi_callerid:
agi_context: default
agi_extension: 123
agi_priority: 2
```

Передав эти переменные, Asterisk посылает пустую строку. Это сигнал того, что Asterisk закончила передачу переменных и сценарий AGI может управлять диалпланом.

На этом этапе сценарий AGI посылает команды в Asterisk, выполняя запись в `STDOUT`. На каждую команду, передаваемую сценарием, Asterisk возвращает ответ, который сценарий AGI должен прочитать. Эти действия (отправка команд в Asterisk и чтение ответов) могут продолжаться в течение всего времени выполнения сценария AGI.

Наверное, вас интересует, какие команды можно использовать в сценарии AGI. Хороший вопрос. Очень скоро мы рассмотрим основные команды[1].

## Вызов сценария AGI из диалплана

Чтобы сценарий AGI работал правильно, он должен быть исполняемым файлом. Для использования сценария AGI в диалплане просто вызывается приложение `AGI()` с указанием имени сценария AGI в качестве аргумента:

```
exten => 123,1,Answer()
exten => 123,2,AGI(agi-test.agi)
```

Сценарии AGI часто располагаются в папке AGI (которая обычно находится в папке /var/lib/asterisk/agi-bin), но можно указать и полный путь к сценарию AGI.

В этой главе сначала мы рассмотрим сценарий agi-test.agi, поставляемый с Asterisk (который написан на Perl), затем напишем AGI-программу на PHP для получения сводки погоды и напоследок создадим математическую игру в виде AGI-программы на Python.

---

[1] Чтобы получить список доступных команд AGI, введите `show agi` в интерфейсе командной строки Asterisk. Также за справкой по командам AGI можно обратиться к приложению C.

### AGI(), EAGI(), DeadAGI() и FastAGI()

Кроме приложения `AGI()`, существует еще несколько AGI-приложений, подходящих для разных ситуаций. Хотя они не будут рассмотрены в данной главе, поняв азы работы со сценариями AGI, разобраться с ними будет довольно просто.

Приложение `EAGI()` (улучшенный AGI) ведет себя так же, как и `AGI()`, но обеспечивает возможность сценарию AGI читать входящий аудиопоток в описатель файла номер три.

Приложение `DeadAGI()` также очень похоже на `AGI()`, но выполняется корректно для «мертвого» канала (то есть канала, который был отключен). Отсюда следует, что обычное приложение `AGI()` не работает для отключенных каналов.

Приложение `FastAGI()` позволяет вызывать сценарий AGI по сети, таким образом, множество серверов Asterisk могут использовать сценарии AGI, хранящиеся централизованно.

## Написание сценариев AGI на Perl

Asterisk поставляется с образцом сценария AGI под названием agi-test.agi. На примере этого файла рассмотрим основные концепции программирования AGI. Этот конкретный сценарий написан на Perl, но AGI-программы могут быть реализованы практически на любом языке программирования. Чтобы доказать это, в данной главе будут представлены AGI-программы на нескольких других языках программирования.

Итак, приступим! Будем рассматривать каждый раздел кода по очереди и описывать, что он делает:

```
#!/usr/bin/perl
```

Эта строка сообщает системе, что данный сценарий написан на Perl, таким образом, для его выполнения должен использоваться интерпретатор Perl. Опытным создателям сценариев для Linux или UNIX эта строка должна быть хорошо знакома. Конечно, здесь предполагается, что исполняемый файл Perl располагается в папке /usr/bin/. Если необходимо, измените строку соответственно местоположению своего интерпретатора Perl.

```
use strict;
```

Строка `use strict` указывает Perl строго придерживаться правил программирования и не допускать возможных ошибок при написании программы, таких как, например, необъявленные переменные. Хотя она не является обязательной, но активация этой функциональности поможет избежать обычных ошибок при программировании.

```
$|=1;
```

Данная строка указывает интерпретатору Perl не буферизировать вывод. Иначе говоря, любые данные должны записываться немедленно, а не накапливаться и выводиться блоками. К этому вопросу мы будем многократно возвращаться по ходу главы.

```
# Задаем некоторые переменные
my %AGI; my $tests = 0; my $fail = 0; my $pass = 0;
```

При написании сценариев AGI должен использоваться только небуферизированный вывод. В противном случае AGI может работать не так, как от него требуется. Например, Asterisk может ожидать вывода программы, тогда как программа полагает, что уже отправила вывод в Asterisk, и ожидает ответа.

Здесь задаются четыре переменные. Первая – это хеш `AGI`, который используется для хранения переменных, передаваемых Asterisk в наш сценарий в начале сеанса AGI. Следующие три – это скалярные значения, используемые для подсчета общего количества тестов, количества непройденных тестов и количества пройденных тестов соответственно:

```
while(<STDIN>) {
   chomp;
   last unless length($_);
   if (/^agi_(\w+)\:\s+(.*)$/) {
      $AGI{$1} = $2;
   }
}
```

Как говорилось ранее, Asterisk передает группу переменных в программу AGI при запуске. Этот цикл просто принимает все эти переменные и сохраняет их в хеше `AGI`. Позже они могут использоваться в программе или просто игнорироваться, но они обязательно должны быть прочитаны из `STDIN`, прежде чем будет продолжено выполнение логики программы.

```
print STDERR "AGI Environment Dump:\n";
foreach my $i (sort keys %AGI) {
   print STDERR " -- $i = $AGI{$i}\n";
}
```

Данный цикл просто записывает каждое из значений, сохраненных в хеше `AGI`, в `STDERR`. Это полезно для отладки сценария AGI, поскольку `STDERR` выводится в консоли Asterisk[1].

```
sub checkresult {
   my ($res) = @_;
   my $retval;
```

[1] На самом деле в консоли Asterisk, вызванной первой (то есть это первый экземпляр Asterisk, вызванный опцией `-c`). Если для запуска Asterisk использовался сценарий `safe_asterisk`, первая консоль Asterisk будет выполняться на TTY9, а это означает, что вы не сможете просматривать ошибки AGI удаленно.

```
$tests++;
chomp $res;
if ($res =~ /^200/) {
   $res =~ /result=(-?\d+)/;
   if (!length($1)) {
      print STDERR "FAIL ($res)\n";
      $fail++;
   } else {
      print STDERR "PASS ($1)\n";
      $pass++;
   }
} else {
   print STDERR "FAIL (unexpected result '$res')\n";
   $fail++;
}
```

Эта подпрограмма считывает результат выполнения команды AGI из Asterisk и декодирует его, чтобы выяснить, была ли команда выполнена успешно или дала сбой.

Теперь, когда подготовительные этапы пройдены, можно перейти к основной логике сценария AGI:

```
print STDERR "1. Testing 'sendfile'...";
print "STREAM FILE beep \"\"\n";
my $result = <STDIN>;
&checkresult($result);
```

Первый тест показывает, как использовать команду `STREAM FILE`. Команда `STREAM FILE` указывает Asterisk воспроизвести звуковой файл вызывающему абоненту, точно так же как это делает приложение `Background()`. В данном случае Asterisk должна воспроизвести файл beep.gsm[1].

Обратите внимание, второй аргумент заменяется парой двойных кавычек, экранированных обратным слэшем. Без обозначения второго аргумента двойными кавычками эта команда не будет работать правильно.

В команды AGI должны передаваться *все необходимые аргументы*. Если требуется пропустить необходимый аргумент, должны быть указаны пустые кавычки (правильно экранированные, соответственно синтаксису конкретного языка программирования), как показано выше. Если необходимое количество аргументов не будет передано, сценарий AGI не станет работать.

Также необходимо убедиться, что вы не забыли передать символы перевода строки (символы `\n` в конце выражения `print`) в конце команды.

После передачи команды `STREAM FILE` этот тест читает результат из `STDIN` и вызывает подпрограмму `checkresult`, чтобы выяснить, смогла ли Asterisk

[1] Asterisk автоматически выбирает лучший формат исходя из затрат на преобразование и доступности, поэтому расширение файла в данной функции никогда не указывается.

воспроизвести файл. Команда STREAM FILE принимает три аргумента, два из которых являются обязательными:

- Имя звукового файла для воспроизведения.
- Коды, которые могут прерывать воспроизведение.
- Место начала воспроизведения звукового файла, заданное номером музыкального фрагмента (необязательный).

Одним словом, этот тест указал Asterisk воспроизвести файл beep.gsm и затем проверил результат, чтобы убедиться, что Asterisk успешно выполнила команду.

```
print STDERR "2. Testing 'sendtext'...";
print "SEND TEXT \"hello world\"\n";
my $result = <STDIN>;
&checkresult($result);
```

Этот тест демонстрирует, как вызывать команду SEND TEXT, которая является аналогом приложения SendText(). Эта команда будет посылать заданный текст вызывающему абоненту, если используемый им тип канала поддерживает передачу текста.

Команда SEND TEXT принимает один аргумент: текст, который должен быть отправлен в канал. Если текст содержит пробелы (как в предыдущем фрагменте кода), аргумент должен быть заключен в кавычки, чтобы Asterisk понимала, что вся строка является одним аргументом команды. Опять же, обратите внимание, что кавычки экранированы, поскольку они должны быть переданы в Asterisk, а не использоваться для ограничения строки в Perl.

```
print STDERR "3. Testing 'sendimage'...";
print "SEND IMAGE asterisk-image\n";
my $result = <STDIN>;
&checkresult($result);
```

Этот тест вызывает команду SEND IMAGE, которая является аналогом приложения SendImage(). Ее единственный аргумент – имя файла изображения, который будет отправляться вызывающему абоненту. Как и команда SEND TEXT, данная команда работает, только если вызывающий канал поддерживает прием изображений.

```
print STDERR "4. Testing 'saynumber'...";
print "SAY NUMBER 192837465 \"\"\n";
my $result = <STDIN>;
&checkresult($result);
```

Этот тест посылает Asterisk команду SAY NUMBER. Она ведет себя аналогично приложению диалплана SayNumber() и принимает два аргумента:

- Число, которое должно быть воспроизведено.
- Коды, которые могут прервать выполнение команды.

Опять же, поскольку второй аргумент опущен, необходимо передать пустую пару кавычек.

```
print STDERR "5. Testing 'waitdtmf'...";
print "WAIT FOR DIGIT 1000\n";
my $result = <STDIN>;
&checkresult($result);
```

Этот тест демонстрирует применение команды WAIT FOR DIGIT. Эта команда обеспечивает ожидание ввода DTMF-кода вызывающим абонентом заданное количество миллисекунд. Если требуется реализовать бесконечное ожидание ввода цифры, в качестве времени ожидания задается -1. Это приложение возвращает десятичное значение ASCII нажатой цифры.

```
print STDERR "6. Testing 'record'...";
print "RECORD FILE testagi gsm 1234 3000\n";
my $result = <STDIN>;
&checkresult($result);
```

Этот фрагмент кода демонстрирует команду RECORD FILE. Она используется для записи разговора, аналогично приложению диалплана Record(). RECORD FILE принимает семь аргументов, из которых три последних являются необязательными:

- Имя записываемого файла.
- Формат, в котором выполняется запись.
- Коды, которые могут прервать запись.
- Время ожидания (максимальное время записи) в миллисекундах или -1, если время ожидания бесконечно.
- Число музыкальных фрагментов, которые необходимо пропустить перед началом записи (необязательный).
- Слово BEEP, если требуется, чтобы Asterisk подавала звуковой сигнал перед началом записи (необязательный).
- Количество секунд, которое должно пройти, прежде чем Asterisk решит, что пользователь закончил запись, и продолжит выполнение даже несмотря на то, что время ожидания еще не истекло и DTMF-коды не были введены (необязательный). Этот аргумент должен следовать за s=.

В данном конкретном случае записывается файл testagi (в формате GSM), любой DTMF-код от 1 до 4 может прервать запись и максимальное время записи – 3000 мс.

```
print STDERR "6a. Testing 'record' playback...";
print "STREAM FILE testagi \"\"\n";
my $result = <STDIN>;
&checkresult($result);
```

Вторая часть этого теста воспроизводит записанный ранее аудиофайл, используя команду STREAM FILE. Команда STREAM FILE уже рассматривалась, поэтому данный фрагмент кода не требует дополнительных пояснений.

```
print STDERR "================= Complete ====================\n";
print STDERR "$tests tests completed, $pass passed, $fail failed\n";
print STDERR "================================================\n";
```

В конце сценария AGI результаты тестирования записываются в STDERR, который должен быть выведен в консоли Asterisk.

Итак, при написании AGI-программ на Perl необходимо помнить следующее:

- Должен быть активирован строгий контроль выполнения правил языка программирования с помощью команды `use strict`[1].
- Должна быть отключена буферизация вывода через задание `$|=1`.
- Данные, поступающие от Asterisk, принимаются с помощью цикла `while(<STDIN>)`.
- Значения записываются командой `print`.
- Для записи отладочной информации в консоль Asterisk используется команда `print STDERR`.

## Библиотека AGI для Perl

Тем, кто собирается создавать собственные сценарии AGI на языке Perl, вероятно, будет интересен модуль на Perl – Asterisk::AGI, написанный Джеймсом Головичем (James Golovich), который можно найти по адресу *http://asterisk.gnuinter.net*. Модуль Asterisk::AGI еще больше упрощает написание сценариев AGI на Perl.

# Создание сценариев AGI на PHP

Мы обещали обсудить несколько языков программирования, поэтому пойдем дальше и рассмотрим, как выглядит сценарий AGI на PHP. Основные правила программирования AGI те же, изменился только язык программирования. В данном примере мы напишем сценарий AGI для загрузки из Интернета сводки погоды и предоставления вызывающему абоненту информации о температуре, направлении и скорости ветра.

```
#!/usr/bin/php -q
<?php
```

Первая строка указывает системе использовать для выполнения сценария интерпретатор PHP. Опция `-q` отключает HTML-сообщения об ошибках. Необходимо убедиться в отсутствии дополнительных строк между первой строкой и открывающим тегом PHP, поскольку это собьет Asterisk с толку.

```
# внесите соответствующие изменения для получения
# данных по интересующему вас городу
# полный список городов США можно найти по адресу
```

[1] Этот совет, пожалуй, распространяется на написание всех программ на Perl, особенно для новичков.

```
# http://www.nws.noaa.gov/data/current_obs/
$weatherURL="http://www.nws.noaa.gov/data/current_obs/KMDQ.xml";
```

Тем самым вы указываете сценарию AGI, где можно получить информацию о текущих погодных условиях. В данном примере предоставляются данные для Хантсвилла, штат Алабама. Вы можете свободно посетить указанный сайт, на котором найдете полный список станций по всем Соединенным Штатам Америки[1].

```
# не допускайте, чтобы этот сценарий выполнялся дольше 60 с
set_time_limit(60);
```

Здесь мы указываем PHP, что данная программа не должна выполняться более 60 с. Таким образом, сценарий будет гарантированно завершен, если по какой-то причине время его выполнения превысит 60 с.

```
# отключить буферизацию вывода
ob_implicit_flush(false);
```

Эта команда отключает буферизацию вывода, то есть все данные будут отправляться в интерфейс AGI немедленно и не станут накапливаться в буфере.

```
# отключите сообщения об ошибках, поскольку, скорее всего,
# они будут пересекаться с сообщениями интерфейса AGI
error_reporting(0);
```

Эта команда отключает все сообщения об ошибках, поскольку они могут пересекаться с сообщениями интерфейса AGI. (Вероятно, полезно будет закомментировать эту строку при тестировании.)

```
# создать описатели файла в случае необходимости
if (!defined('STDIN'))
{
   define('STDIN', fopen('php://stdin', 'r'));
}
if (!defined('STDOUT'))
{
   define('STDOUT', fopen('php://stdout', 'w'));
}
if (!defined('STDERR'))
{
   define('STDERR', fopen('php://stderr', 'w'));
}
```

Этот фрагмент кода гарантирует открытие описателей файла для потоков `STDIN`, `STDOUT` и `STDERR`, которые будут обрабатывать все взаимодействия между Asterisk и нашим сценарием.

---

[1] Приносим извинения читателям, которые живут не в США, за использование сервиса погоды, предоставляющем информацию только о городах США. Если вы сможете найти хороший международный погодный сервис, который предоставляет свои данные в XML, вам не должно составить особого труда изменить этот сценарий AGI для работы с тем конкретным сервисом. Как только мы найдем такой сервис, мы внесем поправки в этот сценарий для будущих изданий данной книги.

```
# извлекаем все переменные AGI из Asterisk
while (!feof(STDIN))
{
   $temp = trim(fgets(STDIN,4096));
   if (($temp == "") || ($temp == "\n"))
   {
      break;
   }
   $s = split(":",$temp);
   $name = str_replace("agi_","",$s[0]);
   $agi[$name] = trim($s[1]);
}
```

Далее считываем все AGI-переменные, передаваемые нам Asterisk. Использование в PHP команды `fgets` для чтения данных из `STDIN` обеспечит сохранение каждой переменной в хеше `$agi`. Эти переменные могли бы использоваться в логике сценария AGI, но в данном примере мы не будем этого делать.

```
# вывести все переменные AGI в целях отладки
foreach($agi as $key=>$value)
{
   fwrite(STDERR,"-- $key = $value\n");
   fflush(STDERR);
}
```

Здесь переменные возвращаются в `STDERR` для целей отладки.

```
# извлечь эту веб-страницу
$weatherPage=file_get_contents($weatherURL);
```

Эта строка кода обеспечивает извлечение XML-файла с сайта National Weather Service (Национальная метеорологическая служба) и помещение его содержимого в переменную `$weatherPage`. Эта переменная будет использована позже для получения необходимых частей сводки погоды.

```
# получить температуру в градусах по Фаренгейту
if (preg_match("/<temp_f>([0-9]+)<\/temp_f>/i",$weatherPage,$matches))
{
   $currentTemp=$matches[1];
}
```

Данный фрагмент кода извлекает данные о температуре (в градусах по Фаренгейту) из сводки погоды с помощью команды `preg_match`. Для получения необходимых данных эта команда использует совместимые с Perl регулярные выражения[1].

```
# получить направление ветра
if (preg_match("/<wind_dir>North<\/wind_dir>/i",$weatherPage))
{
   $currentWindDirection='northerly';
```

[1] Полный справочник по регулярным выражениям – Джеффри Фридл «Регулярные выражения», 3-е издание. – Пер. с англ. – СПб: Символ-Плюс, 2008.

```
}
elseif (preg_match("/<wind_dir>South<\/wind_dir>/i",$weatherPage))
{
   $currentWindDirection='southerly';
}
elseif (preg_match("/<wind_dir>East<\/wind_dir>/i",$weatherPage))
{
   $currentWindDirection='easterly';
}
elseif (preg_match("/<wind_dir>West<\/wind_dir>/i",$weatherPage))
{
   $currentWindDirection='westerly';
}
elseif (preg_match("/<wind_dir>Northwest<\/wind_dir>/i",$weatherPage))
{
   $currentWindDirection='northwesterly';
}
elseif (preg_match("/<wind_dir>Northeast<\/wind_dir>/i",$weatherPage))
{
   $currentWindDirection='northeasterly';
}
elseif (preg_match("/<wind_dir>Southwest<\/wind_dir>/i",$weatherPage))
{
   $currentWindDirection='southwesterly';
}
elseif (preg_match("/<wind_dir>Southeast<\/wind_dir>/i",$weatherPage))
{
   $currentWindDirection='southeasterly';
}
```

**Направление ветра извлекаем посредством команды** `preg_match`**, а полученное значение (заключенное в теги** `wind_dir`**) присваиваем переменной** `$currentWindDirection`**.**

```
# получаем скорость ветра
if (preg_match("/<wind_mph>([0-9.]+)<\/wind_mph>/i",$weatherPage,$matches))
{
   $currentWindSpeed = $matches[1];
}
```

**Наконец получаем текущую скорость ветра и присваиваем ее значение переменной** `$currentWindSpeed`**.**

```
# сообщить вызывающему абоненту текущие погодные условия
if ($currentTemp)
{
   fwrite(STDOUT,"STREAM FILE temperature \"\"\n");
   fflush(STDOUT);
   $result = trim(fgets(STDIN,4096));
   checkresult($result);
   fwrite(STDOUT,"STREAM FILE is \"\"\n");
   fflush(STDOUT);
   $result = trim(fgets(STDIN,4096));
```

```
    checkresult($result);
    fwrite(STDOUT,"SAY NUMBER $currentTemp \"\"\n");
    fflush(STDOUT);
    $result = trim(fgets(STDIN,4096));
    checkresult($result);
    fwrite(STDOUT,"STREAM FILE degrees \"\"\n");
    fflush(STDOUT);
    $result = trim(fgets(STDIN,4096));
    checkresult($result);
    fwrite(STDOUT,"STREAM FILE fahrenheit \"\"\n");
    fflush(STDOUT);
    $result = trim(fgets(STDIN,4096));
    checkresult($result);
}
if ($currentWindDirection && $currentWindSpeed)
{
    fwrite(STDOUT,"STREAM FILE with \"\"\n");
    fflush(STDOUT);
    $result = trim(fgets(STDIN,4096));
    checkresult($result);
    fwrite(STDOUT,"STREAM FILE $currentWindDirection \"\"\n");
    fflush(STDOUT);
    $result = trim(fgets(STDIN,4096));
    checkresult($result);
    fwrite(STDOUT,"STREAM FILE wx/winds \"\"\n");
    fflush(STDOUT);
    $result = trim(fgets(STDIN,4096));
    checkresult($result);
    fwrite(STDOUT,"STREAM FILE at \"\"\n";)
    fflush(STDOUT);
    $result = trim(fgets(STDIN,4096));
    checkresult($result);
    fwrite(STDOUT,"SAY NUMBER $currentWindSpeed \"\"\n");
    fflush(STDOUT);
    $result = trim(fgets(STDIN,4096));
    checkresult($result);
    fwrite($STDOUT,"STREAM FILE miles-per-hour \"\"\n");
    fflush(STDOUT);
    $result = trim(fgets(STDIN,4096));
    checkresult($result);
}
```

Теперь, собрав все необходимые данные, можно отправить AGI-команды в Asterisk (проверяя результаты по ходу), которые доставят информацию о текущих погодных условиях вызывающему абоненту. Это будет реализовано с помощью AGI-команд `STREAM FILE` и `SAY NUMBER`.

Мы говорили об этом раньше, повторим еще раз: при вызове команд AGI в них должны передаваться все необходимые аргументы. В данном случае обе команды, `STREAM FILE` и `SAY NUMBER`, требуют второго аргумента. Передадим пустые кавычки, экранированные символом обратного слэша.

Также следует обратить внимание, что при каждой записи в STDOUT вызывается команда fflush. Вероятно, это лишнее, но не будет вреда в том, чтобы гарантировать немедленную отправку AGI-команды в Asterisk, без буферизации.

```
function checkresult($res)
{
   trim($res);
   if (preg_match('/^200/',$res))
   {
      if (! preg_match('/result=(-?\d+)/',$res,$matches))
      {
         fwrite(STDERR,"FAIL ($res)\n");
         fflush(STDERR);
         return 0;
      }
      else
      {
         fwrite(STDERR,"PASS (".$matches[1].")\n");
         fflush(STDERR);
         return $matches[1];
      }
   }
   else
   {
      fwrite(STDERR,"FAIL (unexpected result '$res')\n");
      fflush(STDERR);
      return -1;
   }
}
```

Назначение функции checkresult аналогично подпрограмме checkresult из нашего примера на Perl. Как следует из ее имени, она проводит проверку результатов, возвращаемых Asterisk, при каждом вызове команды AGI.

```
?>
```

В конце файла располагается закрывающий тег PHP. После закрывающего тега PHP не должно быть никаких пробелов, поскольку это может сбить с толку интерфейс AGI.

Теперь мы уже рассмотрели два разных языка программирования с целью продемонстрировать, что общего в написании сценария AGI на PHP и Perl и чем они отличаются. При создании сценария AGI на PHP помните, что необходимо:

- Запускать PHP с ключом -q; это отключает HTML в сообщениях об ошибках.
- Отключить ограничение по времени или задать для него приемлемое значение (более новые версии PHP автоматически отключают ограничение по времени при запуске PHP из командной строки).
- Отключить буферизацию вывода с помощью команды ob_implicit_flush(false).

- Открыть описатели файла для `STDIN`, `STDOUT` и `STDERR` (в более новых версиях PHP один или более этих описателей файла уже могут быть открыты; в предыдущем фрагменте кода показано, как сделать это красиво для большинства версий PHP).
- Прочитать переменные из `STDIN`, используя функцию `fgets`.
- Использовать функцию `fwrite` для записи данных в `STDOUT` и `STDERR`.
- Всегда вызывать функцию `fflush` после записи в `STDOUT` или `STDERR`.

## Библиотека AGI для PHP

Для более продвинутого программирования AGI на PHP, вероятно, пригодится проект PHPAGI, который можно найти по адресу *http://phpagi.sourceforge.net*. Изначально он был написан Мэттью Ашамом (Matthew Asham) и дорабатывался несколькими членами сообщества разработчиков Asterisk.

# Написание сценариев AGI на Python

Сценарий AGI, который мы напишем на Python, называется «Игра в вычитание». Источником идей для его написания стала программа на Perl, созданная Эдом Гаем (Ed Guy) и представленная им на конференции AstriCon в 2004 году. Эд рассказывал, в какой восторг он пришел от мощи и простоты Asterisk, когда обнаружил, что может написать короткий сценарий на Perl, чтобы помочь своей дочери с математикой.

Поскольку мы уже написали Perl-программу, использующую AGI и Эд создал свою математическую программу на Perl, мы решили заняться реализацией этой задачи на Python!

Итак, разберем наш сценарий на Python:

```
#!/usr/bin/python
```

Данная строка указывает системе выполнять этот сценарий в интерпретаторе Python. Для небольших сценариев в эту строку можно добавить опцию `-u`, что обеспечит выполнение Python в режиме без буферизации. Однако это не рекомендуется для больших или часто используемых сценариев AGI, поскольку может сказаться на производительности системы.

```
import sys
import re
import time
import random
```

Здесь импортируются несколько библиотек, которые будут использоваться в сценарии AGI.

```
# Читаем и игнорируем среду AGI (читать до пустой строки)

env = {}
tests = 0;
```

```
while 1:
   line = sys.stdin.readline().strip()

   if line == '':
      break
   key,data = line.split(':')
   if key[:4] <> 'agi_':
      # игнорируем ввод, который начинается не с agi_
      sys.stderr.write("Did not work!\n");
      sys.stderr.flush()
      continue
   key = key.strip()
   data = data.strip()
   if key <> '':
      env[key] = data

sys.stderr.write("AGI Environment Dump:\n");
sys.stderr.flush()
for key in env.keys():
   sys.stderr.write(" -- %s = %s\n" % (key, env[key]))
   sys.stderr.flush()
```

**Данный фрагмент кода читает переменные, передаваемые в сценарий из Asterisk, и сохраняет их в словарь** `env`**. Затем эти значения записываются в** `STDERR` **для целей отладки.**

```
def checkresult (params):
   params = params.rstrip()
   if re.search('^200',params):
   result = re.search('result=(\d+)',params)
   if (not result):
      sys.stderr.write("FAIL ('%s')\n" % params)
      sys.stderr.flush()
      return -1
   else:
      result = result.group(1)
      #debug("Result:%s Params:%s" % (result, params))
      sys.stderr.write("PASS (%s)\n" % result)
      sys.stderr.flush()
      return result
else:
   sys.stderr.write("FAIL (unexpected result '%s')\n" % params)
   sys.stderr.flush()
   return -2
```

**Функция** `checkresult` **по своему назначению практически идентична подпрограмме** `checkresult` **в примере AGI-сценария на Perl, который рассматривался ранее в этой главе. Она читает результат выполнения команды Asterisk, проводит синтаксический разбор результата и сообщает, была команда выполнена успешно или нет.**

```
def sayit (params):
   sys.stderr.write("STREAM FILE %s \"\"\n" % str(params))
   sys.stderr.flush()
```

```
    sys.stdout.write("STREAM FILE %s \"\"\n" % str(params))
    sys.stdout.flush()
    result = sys.stdin.readline().strip()
    checkresult(result)
```

Функция `sayit` – это просто оболочка для команды `STREAM FILE`.

```
def saynumber (params):
    sys.stderr.write("SAY NUMBER %s \"\"\n" % params)
    sys.stderr.flush()
    sys.stdout.write("SAY NUMBER %s \"\"\n" % params)
    sys.stdout.flush()
    result = sys.stdin.readline().strip()
    checkresult(result)
```

Функция `saynumber` – это просто оболочка для команды `SAY NUMBER`.

```
def getnumber (prompt, timelimit, digcount):
    sys.stderr.write("GET DATA %s %d %d\n" % (prompt, timelimit, digcount))
    sys.stderr.flush()
    sys.stdout.write("GET DATA %s %d %d\n" % (prompt, timelimit, digcount))
    sys.stdout.flush()
    result = sys.stdin.readline().strip()
    result = checkresult(result)
    sys.stderr.write("digits are %s\n" % result)
    sys.stderr.flush()
    if result:
        return result
    else:
        result = -1
```

Функция `getnumber` вызывает команду `GET DATA` для получения DTMF-ввода от вызывающего абонента. Она используется в нашей программе для получения ответов абонента на поставленные задачи по вычитанию.

```
limit=20
digitcount=2
score=0
count=0
ttanswer=5000
```

Здесь выполняется задание исходных значений нескольким переменным, которые будут использоваться в программе.

```
starttime = time.time()
t = time.time() - starttime
```

В этих строках переменной `starttime` задается текущее время, а переменной `t` – начальное значение 0. Переменная `t` будет использоваться для отсчета времени с момента запуска сценария AGI в секундах.

```
sayit("subtraction-game-welcome")
```

Далее, мы рады приветствовать абонента в нашей игре на вычитание.

```
while ( t < 180 ):

    big = random.randint(0,limit+1)
    big += 10
```

```
subt= random.randint(0,big)
ans = big - subt
count += 1

#постановка задачи:
sayit("subtraction-game-next");
saynumber(big);
sayit("minus");
saynumber(subt);
res = getnumber("equals",ttanswer,digitcount);

if (int(res) == ans) :
   score+=1
   sayit("subtraction-game-good");
else :
   sayit("subtraction-game-wrong");
   saynumber(ans);

t = time.time() - starttime
```

Это сердце сценария AGI. При циклическом выполнении данного фрагмента кода абоненту в течение 180 с предлагаются задачи на вычитание. В начале цикла берутся два случайных числа и вычисляется их разность. Затем абоненту предлагается решить эту задачу. Читается ответ абонента. Если ответ неверен, дается правильный ответ.

```
pct = float(score)/float(count)*100;
sys.stderr.write("Percentage correct is %d\n" % pct)
sys.stderr.flush()
sayit("subtraction-game-timesup")
saynumber(score)
sayit("subtraction-game-right")
saynumber(count)
sayit("subtraction-game-pct")
saynumber(pct)
```

После того как абонент закончил решение примеров, ему сообщается, сколько баллов он набрал.

Как видите, при написании сценариев AGI на Python следует помнить такие основные моменты:

- Выходной буфер должен очищаться после каждой записи. Это гарантирует, что AGI-программа не зависнет из-за того, что Asterisk будет ожидать освобождения буфера для записи, а Python – ответа от Asterisk.
- Чтение данных из Asterisk осуществляется с помощью команды `sys.stdin.readline`.
- Запись команд в Asterisk выполняется с помощью команды `sys.stdout.write`. После записи не забывайте вызывать `sys.stdout.flush`.

## Библиотека AGI для Python

Если вы планируете много работать с Python для AGI, вероятно, вам пригодится модуль Python Pyst, созданный Карлом Патлэндом (Karl Putland). Его можно найти по адресу *http://sourceforge.net/projects/pyst*.

# Отладка в AGI

Отладка программ AGI, как и любых других программ, может приводить в уныние. К счастью, при отладке сценариев AGI есть два преимущества. Во-первых, поскольку весь обмен информацией между Asterisk и программой AGI происходит через `STDIN` и `STDOUT` (и конечно, `STDERR`), у вас должно получиться выполнять сценарий AGI непосредственно из операционной системы. Во-вторых, в Asterisk есть удобная команда для отображения всех взаимодействий между ним и сценарием AGI – `agi debug`.

## Отладка из операционной системы

Как упоминалось выше, у вас должно получиться запустить свою программу прямо из операционной системы, чтобы проверить ее поведение. Хитрость здесь в том, чтобы действовать подобно Asterisk, предоставляя сценарию следующее:

- Список переменных и их значений, таких как `agi_test:1`.
- Символы перевода строки (`\n`), указывающие на то, что передача переменных завершена.
- Ответы на каждую из команд AGI, поступающую из вашего сценария AGI. Обычно достаточно ввести `200 response=1`.

При тестировании программы непосредственно из операционной системы, возможно, проще замечать ошибки в ней.

## Использование команды Asterisk agi debug

В интерфейсе командной строки Asterisk есть очень полезная команда для отладки сценариев AGI, которая называется (вполне уместно) `agi debug`. Если ввести в консоли Asterisk `agi debug` и затем запустить AGI-сценарий, вы увидите нечто подобное:

```
-- Executing AGI("Zap/1-1", "temperature.php") in new stack
    -- Launched AGI Script /var/lib/asterisk/agi-bin/temperature.php
AGI Tx >> agi_request: temperature.php
AGI Tx >> agi_channel: Zap/1-1
AGI Tx >> agi_language: en
AGI Tx >> agi_type: Zap
AGI Tx >> agi_uniqueid: 1116732890.8
AGI Tx >> agi_callerid: 101
AGI Tx >> agi_calleridname: Tom Jones
```

```
AGI Tx >> agi_callingpres: 0
AGI Tx >> agi_callingani2: 0
AGI Tx >> agi_callington: 0
AGI Tx >> agi_callingtns: 0
AGI Tx >> agi_dnid: unknown
AGI Tx >> agi_rdnis: unknown
AGI Tx >> agi_context: incoming
AGI Tx >> agi_extension: 141
AGI Tx >> agi_priority: 2
AGI Tx >> agi_enhanced: 0.0
AGI Tx >> agi_accountcode:
AGI Tx >>
AGI Rx << STREAM FILE temperature ""
AGI Tx >> 200 result=0 endpos=6400
AGI Rx << STREAM FILE is ""
AGI Tx >> 200 result=0 endpos=5440
AGI Rx << SAY NUMBER 67 ""
    -- Playing 'digits/60' (language 'en')
    -- Playing 'digits/7' (language 'en')
AGI Tx >> 200 result=0
AGI Rx << STREAM FILE degrees ""
AGI Tx >> 200 result=0 endpos=6720
AGI Rx << STREAM FILE fahrenheit ""
AGI Tx >> 200 result=0 endpos=8000
    -- AGI Script temperature.php completed, returning 0
```

Во время выполнения сценария AGI будут выведены строки трех типов. Первый тип – строки, начинающиеся с `AGI TX >>`. Это строки, которые Asterisk передает в `STDIN` вашей программы. Второй тип – строки, начинающиеся с `AGI RX <<`. Это команды, которые ваша AGI-программа записывает в Asterisk через `STDOUT`. Третий тип – строки, начинающиеся с `--`. Это стандартные сообщения Asterisk, выводимые при выполнении определенных команд.

Чтобы отключить отладку AGI после запуска, просто введите в консоли Asterisk `agi no debug`.

Используя команду `agi debug`, можно увидеть взаимодействие между Asterisk и своей программой, что может быть очень полезным при отладке. Надеемся, эти два совета помогут вам создавать и отлаживать мощные AGI-программы.

# Заключение

AGI для разработчика – это один из наиболее революционных и веских аргументов в пользу Asterisk, а не закрытой узкоспециализированной офисной АТС. Но AGI – это только часть картины. В главе 10 будет рассмотрен другой мощный интерфейс программирования, известный как Asterisk Manager Interface.

# 10

# Интерфейс Asterisk Manager (AMI) и Adhearsion

*Лучше позаботьтесь о том, чтобы все слова ваши были понятны, пристойны и правильно расположены, чтобы каждое предложение и каждый ваш период, затейливый и полнозвучный, с наивозможною и доступною вам простотою и живостью передавали то, что вы хотите сказать; выражайтесь яснее, не запутывая и не затемняя смысла.*

– Мигель де Сервантес, предисловие к книге «Дон Кихот»

## Интерфейс Manager

Asterisk Manager Interface (AMI) – мощный программный интерфейс. Он позволяет внешним программам как управлять, так и контролировать систему Asterisk[1]. Этот интерфейс часто используется для интеграции Asterisk с существующими бизнес-процессами и системами, программным обеспечением CRM (Customer Relationship Management – управление взаимоотношениями с клиентами). Он также может применяться для разнообразных приложений, таких как программы автоматического набора номера и системы click-to-call (звонок-по-щелчку).

Интерфейс Asterisk Manager слушает соединения, устанавливаемые по сетевому порту. Клиентская программа может соединяться с интер-

[1] В противоположность Asterisk Gateway Interface (AGI), который обеспечивает Asterisk возможность запускать внешнюю программу из диалплана. Интерфейсы AGI и AMI во многом дополняют друг друга.

фейсом Asterisk Manager через этот порт, аутентифицироваться и передавать команды в Asterisk. После этого Asterisk будет отвечать на запрос, а также обновлять в клиентской программе информацию о статусе системы.

Чтобы использовать интерфейс Manager, необходимо задать учетную запись в файле /etc/asterisk/manager.conf. Этот файл будет выглядеть примерно так:

```
[general]
enabled = yes
port = 5038
bindaddr = 0.0.0.0

[oreilly]
secret = notvery
;deny=0.0.0.0/0.0.0.0
;permit=209.16.236.73/255.255.255.0
read = system,call,log,verbose,command,agent,user
write = system,call,log,verbose,command,agent,user
```

В разделе `[general]` необходимо активировать сервис, задав параметр `enabled = yes`. Чтобы эти изменения вступили в силу, понадобится перезагрузить интерфейс Manager (команда `module reload manager` из консоли Asterisk). По умолчанию используется TCP-порт 5038.

Далее вы создаете по разделу для каждого пользователя, который будет присылать запрос на аутентификацию в системе. Для пользователя задается имя пользователя в квадратных скобках (`[ ]`), за которым следует пароль этого пользователя (`secret`), все IP-адреса, которым вы желаете запретить (`deny`) доступ, все IP-адреса, которым вы хотите разрешить (`permit`) доступ, и права на чтение (`read`) и запись (`write`) для этого пользователя.

Очень важно понимать, что, кроме незашифрованного пароля и возможности ограничить доступ для определенных IP-адресов, в интерфейсе Manager нет других средств обеспечения безопасности. Если вы запускаете Manager в ненадежной сети (или существуют любые другие сложные требования), для обработки всех соединений с API интерфейса Manager следует использовать замечательный пакет AstManProxy Дэвида Троя (David Troy).

## Подключение к интерфейсу Manager

Важно помнить, что интерфейс Manager создан для использования программами, а не пользователями. Дело здесь не в том, что не получится направлять команды к нему напрямую, просто не следует ожидать увидеть обычный консольный интерфейс – назначение интерфейса Manager не в этом.

Команды в интерфейс Manager доставляются в пакетах, имеющих следующий синтаксис (строки завершаются CR+LF)[1]:

```
Действие: <тип действия>
Ключ 1: Значение 1
Ключ 2: Значение 2
и т. д. ...
Переменная: Значение
Переменная: Значение
и т. д. ...
```

Например, чтобы пройти аутентификацию в интерфейсе Manager (которая необходима для получения возможности любого взаимодействия), необходимо передать следующее:

```
Action: login
Username: oreilly
Secret: notvery
<CR+LF>
```

Дополнительная CR+LF в пустой строке обеспечит передачу в интерфейс Manager пакета целиком.

Пройдя аутентификацию, вы сможете запускать действия, а также видеть события, сформированные Asterisk. В сильно загруженной системе может быть очень сложно или практически невозможно отслеживать все это «невооруженным глазом». Чтобы отключить для Asterisk возможность посылать события, можно добавить параметр Events в команду на регистрацию:

```
Action: login
Username: oreilly
Secret: notvery
Events: off
<CR+LF>
```

Если вы боитесь передавать свой пароль по сети незашифрованным (и это нормально), можно осуществить аутентификацию, используя систему запрос/ответ и алгоритм MD5, принцип работы которого очень похож на краткую аутентификацию HTTP. Для этого сначала вызывается действие Challenge (Запрос), которое предоставит вам маркер запроса:

```
Action: Challenge
AuthType: MD5

Response: Success
Challenge: 840415273
```

[1] Возврат каретки с переводом строки (Carriage Return + Line Feed). Как правило, это обеспечивается нажатием клавиши Enter, но может отличаться для различных платформ ОС и языков программирования, поэтому, если имеются какие-то проблемы с передачей команд в интерфейс, вероятно, нелишним будет точно указать необходимое сочетание клавиш. На момент написания данной книги в Википедии имеется подробное описание этой концепции (*http://en.wikipedia.org/wiki/Newline*).

После этого можно взять маркер запроса, присоединить в конце него незашифрованный секрет и вычислить контрольную сумму результирующей строки по алгоритму MD5. Результат может использоваться для регистрации без необходимости передачи секрета открытым текстом.

```
Action: Login
AuthType: MD5
Username: Admin
Key: e7a056e1488882c6c509bbe71a049978

Response: Success
Message: Authentication accepted
```

## Передача команд

После успешной регистрации в системе AMI можно передавать команды в Asterisk, используя другие действия. Здесь мы продемонстрируем несколько команд, чтобы дать представление о том, как они работают.

### Перенаправление вызова

Действие `Redirect` (Перенаправить) может использоваться для перенаправления вызова. После регистрации необходимо послать такое действие:

```
Action: Redirect
Channel: SIP/John-ae201e78
Context: Lab
Exten: 6001
Priority: 1
ActionID: 2340981650981
```

Каждое действие, передаваемое по интерфейсу Manager, может сопровождаться произвольным значением `ActionID`. Это позволит распознавать, к какому действию относится ответ Asterisk. Настоятельно рекомендуется передавать уникальный `ActionID` с каждой командой AMI.

Этот URL переносит заданный канал в другой добавочный номер и приоритет диалплана. Ответ на это действие такой:

```
Response: Success
ActionID: 2340981650981
Message: Redirect Successful
```

### Чтение конфигурационного файла

Чтобы прочитать конфигурационный файл Asterisk через интерфейс Manager, можно использовать действие `GetConfig`. `GetConfig` возвращает содержимое конфигурационного файла или его часть. Следующая команда извлекает содержимое файла users.conf:

```
Action: GetConfig
Filename: users.conf
ActionID: 9873497149817
```

После этого Asterisk возвращает содержимое файла users.conf. Ответ выглядит так:

```
Response: Success
ActionID: 987397149817
Category-000000: general
Line-000000-000000: fullname=New User
Line-000000-000001: userbase=6000
Line-000000-000002: hasvoicemail=yes
Line-000000-000003: hassip=yes
Line-000000-000004: hasiax=yes
Line-000000-000005: hasmanager=no
Line-000000-000006: callwaiting=yes
Line-000000-000007: threewaycalling=yes
Line-000000-000008: callwaitingcallerid=yes
Line-000000-000009: transfer=yes
Line-000000-000010: canpark=yes
Line-000000-000011: cancallforward=yes
Line-000000-000012: callreturn=yes
Line-000000-000013: callgroup=1
Line-000000-000014: pickupgroup=1
Line-000000-000015: host=dynamic
```

### Обновление конфигурационных файлов

Часто полезно иметь возможность обновлять конфигурационный файл Asterisk через интерфейс Manager. Для обновления одной или более настроек конфигурационного файла используется действие `Update Config`. Например, чтобы удалить из users.conf пользователя под именем 6003, можно использовать следующую команду:

```
Action: UpdateConfig
Filename: users.conf
Reload: yes
SrcFilename: users.conf
DstFilename: users.conf
Action-00000: delcat
Cat-00000: 6003
ActionID: 5298795987243
```

Конечно, мы лишь слегка коснулись возможностей Asterisk Manager Interface и рассмотрели лишь несколько из множества предоставляемых им разнообразных действий. Более подробный список доступных команд приведен в приложении F.

## Flash Operator Panel

Flash Operator Panel (FOP) – один из наиболее популярных примеров, демонстрирующих мощь интерфейса Manager. FOP обеспечивает визу-

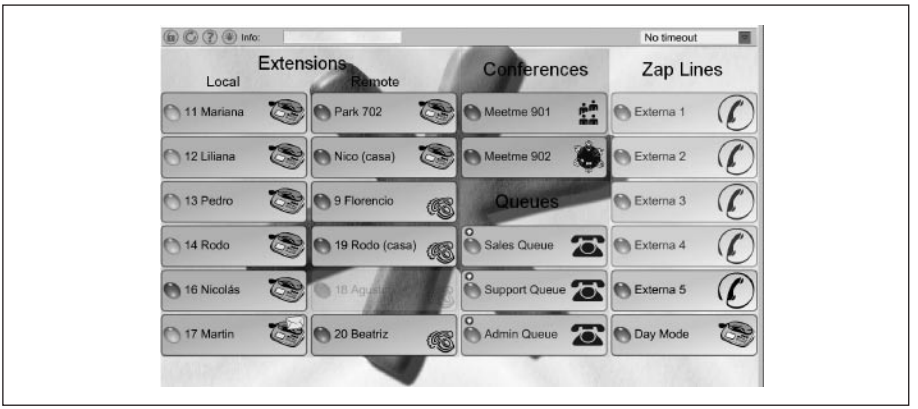



***Рис. 10.1.*** *Интерфейс управления Flash Operator Panel*

альное веб-представление вашей системы и возможность управления вызовами.

FOP чаще всего используется для того, чтобы дать возможность оператору-человеку видеть пользователей системы и устанавливать соединения между ними. Также он может применяться в инфраструктуре центра обработки звонков для обеспечения инициируемых CRM всплывающих экранов[1].

Интерфейс управления FOP представлен на рис. 10.1. Копию FOP можно найти по адресу *http://www.asternic.org*.

Настраивать FOP несложно, но все-таки конфигурация включает несколько этапов. Эти вопросы выходят за рамки рассмотрения данной книги, но на веб-сайте, посвященном FOP, можно найти самую свежую документацию с подробным описанием процесса установки и настройки.

FOP имеет фантастическое сообщество разработчиков и пользующуюся большой популярностью рассылку. Успеху FOP также способствовало его включение в Trixbox.

## Разработка в Asterisk с использованием Adhearsion

Не так давно появилась новая технология, которая может изменить порядок составления диалпланов[2].

---

[1] Customer Relationship Management (CRM) – это интерфейс, используемый компаниями для помощи в управлении информацией и взаимодействиями клиентов.

[2] Мы хотим поблагодарить Джея Филлипса (Jay Phillips) за предоставление идей и кода для данного раздела книги.

## Новый подход к диалпланам

Asterisk повзрослела с точки зрения как технологии, так и ее популярности, но при все большем погружении в этот чудесный мир невозможно не столкнуться с ограничениями. Созданию сложных сценариев уровня предприятия с использованием только Asterisk будет сопутствовать множество трудностей применения логики диалплана. Несмотря на всю гибкость и мощь диалплана, как язык программирования он довольно слаб и существенно менее гибок, чем большинство современных языков сценариев. При реализации расширенной логики диалплан, GUI и даже более развитый AEL (Asterisk Extension Language) могут разочаровать.

Создавая все более сложные диалпланы, можно столкнуться с трудностями в следующих вопросах:

- Условные циклы и переходы по условию.
- Переменные.
- Сложные структуры данных.
- Интеграция с базой данных/LDAP.
- Использование библиотек сторонних производителей.
- Обмен и распространение функциональности VoIP.
- Расширение конфигурационных языков.
- Плохая обработка ошибок.
- Плохая обработка даты и времени.
- Сопоставление с шаблонами.
- Единообразие использования.
- Организация исходного кода.

Многие решают эти проблемы, реализуя расширенную логику во внешних программах на таких языках программирования, как Perl и PHP, и соединяясь с Asterisk через AMI и AGI. К сожалению, обеспечивая желаемую мощь, эти решения не всегда упрощают жизнь разработчика. Наоборот, они зачастую еще более усложняют процесс разработки. Используя существующие технологии в Asterisk, но имея целью обеспечение мощи и простоты, Adhearsion предлагает новый подход.

## Разработка в Asterisk с использованием Adhearsion

Adhearsion – это инфраструктура с открытым исходным кодом (распространяемая по лицензии LGPL), которая разработана с целью улучшения реализации решений Asterisk. Она располагается поверх системы Asterisk, обрабатывая части или весь диалплан, и обеспечивает несколько уникальных способов управления доступом к Asterisk посредством ряда улучшенных интерфейсов. Adhearsion выполняется в отдельном процессе-демоне и интегрируется через уже представленные выше интерфейсы Gateway (AGI) и Manager (AMI), поэтому конфигурирование контекста на ее использование заключается просто

в добавлении нескольких строк в диалплан или пользователя в файл manager.conf.

Adhearsion преимущественно использует высокодинамичный объектно-ориентированный язык программирования Ruby, но может поддерживать другие языки, такие как C или Java. В мире VoIP многие вещи существуют как концептуальные объекты, то есть применение объектно-ориентированного программирования вполне логично. У тех, кто хорошо знаком с Python, Perl или другими языками сценариев, не должно возникнуть проблем с Ruby. Для тех, кто не работал до этого с языками сценариев, Ruby – замечательное начало.

## Установка Adhearsion

Программное обеспечение Ruby, как правило, устанавливается с помощью диспетчера пакетов (аналогичного диспетчерам пакетов Linux, но созданного специально для платформы Ruby). Adhearsion входит в стандарт RubyGems, поэтому, если установлены Ruby и RubyGems, вы находитесь на расстоянии одной команды от установки Adhearsion.

### Установка Ruby/RubyGems в AsteriskNOW

AsteriskNOW стандартно поставляется с Ruby, но без RubyGems (по причинам поддержки). К счастью, RubyGems можно без труда установить из коллекции Ruby rPath, используя следующую команду:

```
conary update rubygems=ruby.rpath.org@rpl:devel
source /etc/profile
```

### Установка Ruby/RubyGems в Linux

Диспетчеры пакетов многих дистрибутивов Linux содержат пакет Ruby, хотя в некоторых до сих пор нет RubyGems. В предпочтительном приложении управления разработкой и сопровождением ПО своего дистрибутива установите Ruby 1.8.5 или более позднюю версию и RubyGems, если он доступен. Если RubyGems недоступен в CentOS, Ruby можно установить, введя следующее:

```
yum install ruby
```

Далее необходим RubyGems. Чтобы получить его, перейдите в /usr/src/ и введите следующее:

```
wget http://rubyforge.org/frs/download.php/20585/rubygems-0.9.3.tgz
tar zxvf rubygems-0.9.3.tgz
cd rubygems-0.9.3
ruby setup.rb
```

### Установка Ruby/RubyGems в Mac OS X

Фактически Ruby поставляется с OS X, но понадобится обновить его и установить RubyGems с MacPorts, диспетчера пакетов OS X. Если

MacPorts установлен (доступен на *http://www.macports.org*, если у вас его еще нет), Ruby и RubyGems можно установить, используя следующую команду:

```
sudo port install ruby rb-rubygems
```

Также может понадобиться добавить `/opt/local/bin` в переменную `PATH` в `/etc/profile`.

### Ruby/RubyGems в Windows

Для Windows существует замечательная программа установки «одним щелчком». Этот инсталлятор за несколько минут автоматически установит Ruby, RubyGems и несколько других обычно используемых пакетов. Инсталлятор можно скачать по адресу *http://rubyforge.org/projects/rubyinstaller*.

### Установка Adhearsion из RubyGems

Выполнив для своей системы все представленные выше инструкции и получив Ruby и RubyGems, установите Adhearsion, используя следующую команду:

```
gem install adhearsion
```

Если обнаружены какие-либо зависимости, вероятно, необходимо разрешить их установку, чтобы обеспечить нормальную работу Adhearsion.

## Создание нового проекта Adhearsion

После того как Adhearsion установлена, можно приступать к созданию и запуску нового проекта Adhearsion с использованием новой команды `ahn`, утилиты командной строки, которая выполняет в Adhearsion практически все.

Вот пример команды для создания нового проекта Adhearsion:

```
ahn create ~/новыйпроект
```

При этом в заданной папке создается подпапка, содержащая папку и иерархию файлов, необходимые для работы Adhearsion. У вас сразу же должно получиться запустить новое приложение по команде

```
ahn start ~/новыйпроект
```

Чтобы ознакомиться с системой Adhearsion, просмотрите папки приложения и прочитайте сопутствующую документацию.

### Написание диалплана в Adhearsion

Как правило, новички начинают с использования возможности написания диалпланов в Adhearsion. Поскольку Ruby допускает тонкую настройку самого языка во время выполнения, одна из функций, выполняемых Adhearsion, – реализация эстетических изменений, которые призваны упростить процесс разработки диалпланов.

Ниже приведено приложение Hello World (Здравствуй, мир), написанное в Adhearsion:

```
мой_первый_контекст {
   play "hello-world"
}
```

Это абсолютно допустимый синтаксис Ruby, но не все приложения Ruby так выглядят. В Adhearsion очень удобно объявлять имена контекстов, поскольку сценарий диалплана здесь обрабатывается особым образом. Ваши сценарии будут располагаться в корневой папке вновь созданного приложения Adhearsion.

Когда вызовы поступают в Asterisk и потом в Adhearsion, Adhearsion вызывает собственную версию имени контекста, из которого происходит AGI-запрос. Таким образом, имя контекста в файле extensions.conf должно гарантированно быть таким же, чтобы обеспечить соответствующее перенаправление вызовов в Adhearsion.

Синтаксис направления вызовов в Adhearsion следующий:

```
[мой_первый_контекст]
exten => _.,1,AGI(agi://127.0.0.1)
```

Любой набранный шаблон фиксируется здесь и отправляется в Adhearsion через AGI для реализации инструкций по обработке вызова. Предоставленный здесь IP, конечно, должен быть заменен на IP, который обеспечит доступ к вашему серверу Adhearsion.

Теперь, имея базовое понимание того, как взаимодействуют Adhearsion и Asterisk, можно перейти к более реалистичному примеру диалплана в Adhearsion:

```
internal {
   case extension
when 10..99
   dial SIP/extension
when 6000..6020, 7000..7030
   # Присоединяемся к конференции MeetMe с помощью join
   join extension
when _'21XX'
   if Time.now.hour.between? 2, 10
      dial SIP/"berlin-office"/extension[2..4]
   else speak "The German office is closed"
   end
when US_NUMBER
   dial SIP/'us-trunk-out'/extension
when /^\d{11,}$/ # Perl-подобное регулярное выражение
   # Передаем все остальные длинные номера прямо в наш
   # магистральный канал связи.
   dial IAX/'intl-trunk-out'/extension
else
   play %w'sorry invalid extension please-try-again'
   end
}
```

Такой небольшой фрагмент кода реализует довольно многое. Даже имея ограниченные познания в Ruby или не зная его вовсе, вы поймете следующее:

- Переменная `extension` (которую Adhearsion создает для нас) используется в условном выражении.
- Набор номера от 10 до 99 направляет нас к равноправному участнику SIP с соответствующим числовым именем пользователя.
- Любой номер в диапазоне от 6000 до 6200 или от 7000 до 7030 направляется в конференцию MeetMe под тем же номером. Конечно, для этого требуется, чтобы номера данных конференций были сконфигурированы в meetme.conf.
- Опция `_'21XX'` точно отвечает стилю шаблонов Asteris. Начало строки с символа подчеркивания в Adhearsion обеспечивает неявный вызов метода, который возвращает регулярное выражение Ruby. В Ruby-выражении `case` регулярные выражения могут использоваться в выражении `when` для выполнения сопоставления с шаблоном. Конечный результат должен быть очень хорошо знаком тем, кто имеет опыт написания extensions.conf.
- Синтаксис Adhearsion для представления каналов также происходит непосредственно из традиционного формата Asterisk. `SIP/123` может использоваться как есть для представления равноправного участника SIP 123. Если бы использовался магистральный канал, синтаксис был бы такой: `SIP/имяканала/имяпользователя`.
- Метод `speak()` обобщает лежащий в основе механизм преобразования текста в речь. Он может быть сконфигурирован на использование наиболее популярных механизмов.
- В выражении `when` для осуществления более сложного сопоставления с шаблонами, если шаблонов Asterisk недостаточно, может использоваться полноценное Perl-подобное регулярное выражение.
- Adhearsion определяет несколько констант, которые могут быть полезны при написании диалпланов. Константа `US_NUMBER` здесь – это регулярное выражение, соответствующее телефонному номеру в США.
- Если необходимо воспроизводить несколько файлов последовательно, `play()` принимает массив имен файлов. К счастью, в Ruby есть удобный способ создания массива строковых значений (String).

Конечно, это только простой пример, демонстрирующий лишь самые основы возможностей Adhearsion для создания диалплана.

## Интеграция с базами данных

Несмотря на чрезвычайный успех в области веб-разработки для обслуживания динамического содержимого, интеграция с базами данных всегда была и остается недостаточно реализованной возможностью управления динамическими голосовыми приложениями в Asterisk. Большинство приложений Asterisk, выполняющих интеграцию с базами данных, делегируют реализацию сложных вопросов AGI-сценариям на

PHP или Perl, потому что extensions.conf или синтаксиса AEL просто недостаточно для решения задач такого уровня сложности.

Adhearsion использует библиотеку интеграции с базами данных ActiveRecord, разработанную создателями инфраструктуры Ruby on Rails. Имея ActiveRecord, конечный пользователь изредка, если вообще делает это, пишет SQL-выражения. А разработчик осуществляет доступ к базе данных, как к любому другому объекту Ruby. Благодаря обеспечиваемым Ruby гибкости и динамичности, доступ к базе данных выглядит и ощущается довольно естественным. Кроме того, ActiveRecord устраняет различия между системами управления базами данных, делая реализацию доступа к базе данных универсальной.

Не вдаваясь в детали ActiveRecord и более сложные варианты ее использования, рассмотрим следующую простую схему MySQL:

```
CREATE TABLE groups (
  `id` int(11) DEFAULT NULL auto_increment PRIMARY KEY,
  `description` varchar(255) DEFAULT NULL,
  `hourly_rate` decimal DEFAULT NULL
);

CREATE TABLE customers (
  `id` int(11) DEFAULT NULL auto_increment PRIMARY KEY,
  `name` varchar(255) DEFAULT NULL,
  `phone_number` varchar(10) DEFAULT NULL,
  `usage_this_month` int(11) DEFAULT 0,
  `group_id` int(11) DEFAULT NULL
);
```

В реальности, конечно, о заказчике хранилось бы намного больше сведений и информация об использовании сервиса находилась бы в записи параметров вызова в базе данных, но такое упрощение позволяет более эффективно продемонстрировать основные моменты ActiveRecord.

Чтобы подключить Adhearsion к этой базе данных, необходимо просто задать информацию для доступа к базе данных в конфигурационном файле YAML:

```
adapter: mysql
host: localhost
database: adhearsion
username: root
password: pass
```

Так Adhearsion будет знать, как подключиться к базе данных, однако механизм доступа к информации в таблицах зависит от того, как смоделированы наши объекты ActiveRecord. Поскольку объект – это экземпляр класса, для каждой таблицы ставим в соответствие класс. С помощью методов надкласса определяем простые свойства и отношения в классе.

Вот два класса, которые могут использоваться с вышеупомянутыми таблицами:

```
class Customer < ActiveRecord::Base
   belongs_to :group

   validates_presence_of :name, :phone_number
   validates_uniqueness_of :phone_number
   validates_associated :group
   def total_bill
self.group.hourly_rate * self.usage_this_month / 1.hour
   end
end

class Group < ActiveRecord::Base
   has_many :customers
   validates_presence_of :description, :hourly_rate
end
```

Уже из этого небольшого объема информации ActiveRecord может сделать множество логических выводов. При обработке данных классов ActiveRecord переводит их имена в нижний регистр, ставит во множественное число и принимает, что это – имена таблиц (`customers` и `groups` соответственно). Если применение такого соглашения нежелательно, автор может без труда переопределить его. Кроме того, во время интерпретации ActiveRecord на самом деле заглядывает в столбцы базы данных и делает доступными многие новые создаваемые динамически методы.

Методы `belongs_to` (принадлежит) и `has_many` (имеет много) в данном примере определяют отношения между `Customers` (клиенты) и `Groups` (группы). Опять обратите внимание, как ActiveRecord использует множественное число в строке `has_many :customers` для большей выразительности кода. В этом примере также можно увидеть несколько проверок достоверности – политик, которые будет применять ActiveRecord. При создании нового объекта `Customer` мы должны обеспечить для него как минимум `name` (имя) и `phone_number` (номер телефона). Задание двух телефонных номеров может привести к конфликту. У каждого `Customer` должна быть `Group`. У каждой группы должно быть `description` (описание) и `hourly_rate` (почасовая ставка). Это поможет разработчику избежать ошибок, а также нарушения целостности базы данных.

Также обратите внимание на метод `total_bill` (общий счет) класса `Customer`. Для любого объекта `Customer`, извлекаемого из базы данных, можно вызвать этот метод, который умножает значение `hourly_rate` для группы, к которой принадлежит `Customer`, на время пользования телефоном этого клиента (в секундах).

Вот несколько примеров, которые могут точнее продемонстрировать то, насколько удобно применять абстрактную объектную логику Ruby для работы с базами данных:

```
everyone = Customer.find :all

jay = Customer.find_by_name "Jay Phillips"
jay.phone_number # Выполняем выражение SELECT
jay.total_bill # Выполняем вычисления по нескольким выражениям SELECT
```

```
jay.group.customers.average :usage_this_month

jay.group.destroy
jay.group = Group.create :description => "New cool group!",
                         :hourly_rate => 1.23
jay.save
```

**Интеграция с базой данных стала здесь намного более естественной, а диалпланы Asterisk выглядят более выразительными. Ниже представлен пример диалплана поставщика сервисов, который налагает ограничение на суммарную продолжительность исходящих звонков, используя информацию из базы данных. Постараемся сохранить простоту:**

```
# Предположим, сервис VoIP предлагается клиентам,
# которые могут быть идентифицированы по их callerid.

service {
   # Строка кода ниже реализует выражение SQL SELECT
   # по отношению к нашей базе данных. Метод
   # find_by_phone_number() был создан автоматически,
   # потому что ActiveRecord обнаружила в базе данных
   # столбец phone_number. Adhearsion создает для нас
   # переменную callerid.
   caller = Customer.find_by_phone_number callerid

   usage = caller.usage_this_month
   if usage >= 100.hours
      play "sorry-cant-let-you-do-that"
   else
      play %w'to-hear-your-account-balance press-1
         otherwise wait-moment'
      choice = wait_for_digit 3.seconds

      p choice
      if choice == 1
         charge = usage / 60.0 * caller.group.hourly_rate
         play %W"your-account will-reflect-charge-of
                 $#{charge} this month for #{usage / 60}
                 minutes and #{usage % 60} seconds"
      end

      # Мы также можем записать значение свойства
      # usage_this_month объекта caller. По завершении
      # выполнения метода time новое значение для этого
      # абонента будет внесено в базу данных.
      caller.usage_this_month += time do
         # Засекаем время выполнения данного фрагмента кода.
         dial IAX/'main-trunk'/extension
      end
      caller.save
   end
}
```

Надежная интеграция с базой данных, которую обеспечивает Adhearsion, упрощает управление и разработку для офисной АТС. Хранящаяся централизованно информация позволяет Asterisk напрямую интегрироваться с другими сервисами, обеспечивая при этом более ценные сервисы, которые не могут быть реализованы в рамках традиционных технологий разработки в Asterisk.

## Распространение и повторное использование кода

Приложение Adhearsion располагается в одной папке, поэтому полностью скопировать VoIP-приложение – не сложнее, чем архивировать файлы. Наверное, впервые в сообществе Asterisk разработчики могут запросто обмениваться друг с другом удачными приложениями и дорабатывать их. Кстати, весьма поощряется предоставление кода собственных приложений Adhearsion.

Кроме того, на локальном уровне расширения инфраструктуры Adhearsion, называемые помощниками, могут использоваться повторно или создаваться самостоятельно. Помощниками могут быть как целые вспомогательные инфраструктуры, такие как Micromenus для интеграции с телефонными микроброузерами, так и обычный новый метод диалплана, который возвращает выбираемые случайным образом цитаты Оскара Уайльда.

Ниже представлен простой помощник Adhearsion, написанный на Ruby. Он создает новый метод, который будет существовать во всей инфраструктуре, включая диалплан. В целях сохранения простоты метод загружает XML-документ по заданному URL HTTP и преобразует его в Ruby-объект `Hash` (тип ассоциативного массива Ruby):

```
def remote_parse url
   Hash.from_xml open(url).read
end
```

Заметьте, что вспомогательный файл может включать только эти три строки. При загрузке Adhearsion выполняет сценарий таким образом, что все описанные методы или классы становятся доступными во всей системе.

Для некоторых задач, в частности задач масштабирования Adhearsion, может потребоваться обеспечить в узких местах эффективность короля производительности: языка программирования C. Ниже представлен пример помощника Adhearsion, который возвращает факториал заданного числа:

```
int fast_factorial(int input) {
   int fact = 1, count = 1;
   while(count <= input) {
      fact *= count++;
   }
   return fact;
}
```

Опять же, приведенный здесь код может составлять все содержимое вспомогательного файла. В данном случае, поскольку код написан на C, файл должен называться factorial.alien.c. Это указывает Adhearsion запустить алгоритм для чтения файла, добавить стандартные заголовки разработки языков C и Ruby, скомпилировать файл, кэшировать общий объект, загрузить его в интерпретатор и затем создать для C-метода оболочку на Ruby. Ниже представлен диалплан, который просто воспроизводит факториал шести, используя этот помощник на C:

```
fast_test {
   num = fast_factorial 6
   play num
}
```

Заметьте, что C-метод становится первоклассным методом на Ruby. Числовые объекты Ruby, преданные в метод, преобразуются в элементарный тип C `int`, а затем возвращаемое значение преобразуется обратно в числовой объект Ruby.

Вспомогательные файлы предлагают простой, но надежный способ расширения инструментария VoIP-специалиста. Но главное – полезными помощниками можно обмениваться, отчего выиграет все сообщество.

## Интеграция с настольным телефоном с использованием Micromenus

В условиях все возрастающей конкуренции между производителями современных настольных телефонов, поддерживающих IP, появление микроброузера прошло относительно незамеченным и используется он крайне редко. Принцип прост: физические настольные телефоны создают интерактивные меню, принимая XML по HTTP или что-то подобное. Однако конфликт интересов сыграл с этой технологией злую шутку: каждый производитель создает собственный XML, микроброузеры часто имеют какие-то странности и доступные возможности сильно разнятся.

Инфраструктура Micromenus (Микроменю) существует как помощник Adhearsion и предназначена устранять различия между телефонами разных производителей. В этой очень специальной области разработки (то есть создании меню) для четкой реализации логики независимо от марок телефонов Micromenus использует очень простой основанный на Ruby «доменный язык».

Вот простой пример Micromenu:

```
image 'company-logo'
item "Call an Employee" do
   # Создаем список сотрудников из активных ссылок в базе данных.
   Employee.find(:all).each do |someone|
      # Просто выбираем кого-то, чтобы позвонить ему по телефону.
      call someone.extension, someone.full_name
   end
end
```

```
item "Weather Information" do
   call "Hear the weather report" do
      play weather_report("Portland, OR")
   end
   item "Current: " + weather("Portland, OR")[:current][:temp]
end
item "System Uptime: " + `uptime`
```

Список `item` (элемент) отображается двумя способами. Если дается только строка текста (String), Micromenus формирует лишь текстовый элемент. Если аргументы содержат блок `do/end` вложенной информации, этот текст становится ссылкой на подстраницу, которая формирует визуальное представление этого вложенного содержимого.

Элемент `call` (вызов) также имеет два варианта использования, каждый из них формирует ссылку, которая при выборе инициирует вызов. Когда `call` получает блок `do/end`, он моделирует физический набор номера, заданного в качестве первого аргумента. Если блок `do/end` существует и все вызовы направляются через Adhearsion, выбор этого элемента обеспечивает выполнение функциональности диалплана в рамках блока. Это замечательный пример того, как выгодно выполнять обработку диалпланов и экранных микроброузеров в одной инфраструктуре.

Из этого примера можно сделать еще некоторые четкие выводы о Micromenus:

- Micromenus поддерживает отправку изображений. Если запрашивающий телефон не поддерживает изображения, в ответе они не будут упоминаться.
- Все помощники Adhearsion работают и здесь. В данном примере использовался помощник сводки погоды.
- Вспомогательная инфраструктура Micromenus может использовать интеграцию Adhearsion с базой данных.
- Ruby может выполнять команду, заключенную в открывающие кавычки, и возвращать результат как строку (String). В нашем случае возвращается время безотказной работы (`uptime`).

Конечно, этот пример предполагает, что вы сконфигурировали интеграцию своего приложения с базой данных соответствующим образом и имеете класс `Employee`, соответствующий таблице со столбцами `extension` (расширение) и `full_name` (полное имя).

Поскольку Micromenus просто формирует визуальное представление различных ответов, поступающих по HTTP, обычный веб-броузер тоже может прислать запрос на сервер Micromenus. Для таких более высокотехнологичных конечных точек Micromenus формирует красивый интерфейс с загрузкой Ajax, эффектами DHTML и окнами, которые можно перетаскивать.

Micromenus – это еще одна возможность сделать ваши Adhearsion-приложения для VoIP более надежными.

## Интеграция с веб-приложением

Хотя Adhearsion конструктивно может интегрироваться с практически любым приложением, включая PHP или сервлеты Java, Ruby на платформе Rails является особенно мощным партнером. Rails – среда разработки веб-приложений, широко обсуждаемая в прессе в последнее время, причем абсолютно заслуженно. Ее разработчики используют весь потенциал Ruby, демонстрируя, как метапрограммирование действительно избавляет разработчика от ненужной работы. Поразительная ясность кода Rails и использование принципа Don't Repeat Yourself (Не повторяйся), или DRY, вдохновили дизайнеров на создание Adhearsion в том виде, в каком она существует сейчас.

Начиная с версии 0.8.0, каталог приложений Adhearsion располагается поверх существующего Rails-приложения, совместное использование данных происходит автоматически. Если требуется разработать сложный веб-интерфейс для функциональности VoIP, обратите внимание на этот сногсшибательный дуэт.

## Использование Java

Весь мир пришел в изумление, когда Sun объявила о приеме на работу двух основных специалистов проекта по разработке интерпретатора JRuby Чарльза Наттера (Charles Nutter) и Томаса Энебо (Thomas Enebo) в сентябре 2006 года. JRuby – это интерпретатор Ruby, написанный не на C, а на Java. Поскольку JRuby может компилировать части приложения на Ruby в байт-код Java, JRuby фактически превосходит C-реализацию Ruby 1.8 по многим параметрам и обещает полностью обойти последнюю в ближайшем будущем.

Приложение на Ruby, выполняемое в JRuby, имеет преимущество использования не только библиотек Ruby, но и любых библиотек Java. Выполнение Adhearsion в JRuby обеспечивает потрясающий ассортимент библиотек Java сторонних производителей для написания диалплана офисной АТС. Если окружение вашей компании требует тесной интеграции с другими технологиями Java, внедрение Adhearsion в стек J2EE может предложить необходимую гибкость.

## Дополнительная информация

Больше информации о быстро развивающемся сообществе разработчиков Adhearsion, включая полные разборы примеров по шагам, можно найти на официальном веб-сайте Adhearsion по адресу *http://adhearsion.com*, в официальном блоге, посвященном Adhearsion, по адресу *http://blog.adhearsion.com*, на веб-сайте консалтинговой компании-учредителя Adhearsion по адресу *http://codemecca.com*. За помощью в изучении Ruby обращайтесь на сайт *http://jicksta.com*.

# 11

# Инфраструктура Asterisk GUI

*...Я конструировал маяк, в то время*
*как все остальные строили корабли.*
– Чарльз Саймик

В данной главе представлены компоненты, которые составляют графический пользовательский интерфейс (GUI) и помогают работать с Asterisk. Для тех, кто не использует дистрибутив AsteriskNOW, здесь приводится установка веб-сервера и компонентов GUI. Показано, как настраивать GUI в соответствии со своими задачами. Также предоставлена техническая информация, чтобы разработчики, желающие создать собственный GUI или приложение, могли использовать веб-сервер и компоненты GUI. Мы выражаем благодарность сотрудникам Digium, помогавшим писать эту главу, и особое спасибо за примеры кода, которые они разработали и протестировали.

## Зачем нужен GUI для Asterisk

Asterisk всегда была телефонной системой для смелых. Раньше от нас требовались недюжинные усилия и больше чем просто упорство, чтобы заставить Asterisk выполнять свои распоряжения. Те, кто в своем стремлении освоить ее принимался за конфигурационные файлы и боролся за свои звонки, были вознаграждены, получив мощную и гибкую систему телефонной связи (а также пользующийся большим спросом набор навыков). Однако массовый потребительский рынок не был и до сих пор не готов к написанию сценариев для обработки добавоч-

ных номеров, управлению равноправными участниками сети и решению других задач, составляющих суть администрирования Asterisk.

Еще с самых ранних времен, до версии 1.0, люди пытаются приручить могучую систему Asterisk с помощью генераторов конфигурационных файлов, связанных с базами данных и управляемых посредством различных графических пользовательских интерфейсов (Graphical User Interfaces, GUI). Самым успешным удалось создать приложение, основанное на Asterisk, но ни один из них не обеспечил полной гибкости, которую предлагает самостоятельная среда для написания сценариев. После замены цифрового хокку диалплана на ограниченный список опций полученная система превратилась из собственно Asterisk в систему, основанную на Asterisk. Неплохо, но не достаточно[1].

Чтобы GUI был именно *GUI Asterisk*, в нем должны быть сохранены создаваемые вручную конфигурационные файлы, которые являются *лингва-франка* Asterisk испокон веков. Он должен предоставлять простые графические средства конфигурации без оказания воздействия на базовое программное обеспечение Asterisk или жесткой фиксации решений, которые должны оставаться открытыми для конечного пользователя. Также он должен обеспечивать расширенную функциональность, не загружая компьютер и не захватывая ценные ресурсы, необходимые для выполнения основной задачи – обработки вызовов.

Одновременно с выходом версии Asterisk 1.4 Digium начала разработку проекта Asterisk GUI. Изначально GUI задумывался как компонент встроенного устройства Asterisk от Digium. Устройство, продаваемое и как Asterisk Appliance Developers Kit (AADK – комплект для разработчиков устройств Asterisk), и как самостоятельная конфигурация, представляет собой небольшой полупроводниковый компьютер с необязательными аналоговыми (и в перспективе цифровыми) интерфейсами. GUI был создан с использованием гибкой и расширяемой инфраструктуры, которая переносит максимум задач по отображению и логику проверки достоверности на компьютер клиента. Также была учтена необходимость сохранения возможности создания конфигурационных файлов вручную, но при этом предоставлены автоматизированные средства их редактирования. Созданная в результате инфраструктура получила название AJAM (обыгрывается название популярной технологии Web 2.0 Ajax), что является аббревиатурой от Asynchronous JavaScript and Asterisk Manager (Асинхронный диспетчер JavaScript и Asterisk). Основной AJAM-код, наборы поддерживающих AJAM веб-страниц и расширение диспетчера Asterisk – все вместе, взаимодействуя, формируют инфраструктуру Asterisk GUI.

---

[1] Кстати, два автора данной книги однажды попытались написать идеальный GUI Asterisk. К счастью для вас, они отказались от этого проекта и занялись документацией Asterisk!

# Что такое GUI

Asterisk GUI – это интерфейс, который поставляется с дистрибутивом AsteriskNOW или может быть добавлен в существующую установку Asterisk. Стандартный интерфейс ориентирован на пользователя, желающего применять Asterisk как офисную АТС для небольшого предприятия с довольно типовыми требованиями к системе телефонной связи. Это самое простое, что можно сделать с помощью AJAM; рассматривайте его как бета-интерфейс, который, как можно ожидать, будет развиваться согласно желаниям сообщества. Его появление вызвало большое воодушевление в сообществе разработчиков Asterisk, потому что лежащие в основе GUI технологии поднимают планку возможностей интерфейса офисной АТС. Он также позволяет создавать пользовательские интерфейсы, нацеленные на собственные уникальные требования.

## Марк Спенсер о GUI

Asterisk – мощная платформа для телефонии. Однако ценность этой мощи определяется тем, насколько она может быть полезна конкретным целевым пользователем. Графические интерфейсы (GUI) очень нужны Asterisk. Большинство GUI специально разработаны для определенной задачи. Например, некоторые GUI созданы специально для систем голосовой почты. Другие ориентированы на гостиничную отрасль. Необходимость в универсальном GUI для Asterisk существует, но приходится идти на естественный компромисс между удобством использования и простотой GUI и количеством предлагаемых им функций. Например, GUI, нужный администратору сложных и многофункциональных систем, скорее всего, будет отличаться от того, который требуется администратору офиса, отвечающему только за простые перемещения, добавление записей и изменение системы. Исходя из такого широкого диапазона требований Digium разработала инфраструктуру GUI, названную (неизобретательно) Asterisk GUI. Digium не стала разрабатывать единый GUI, а вместо этого выпустила разные GUI и инфраструктуру для упрощения процесса создания и изменения GUI для разных областей применения.

Второй задачей было обеспечить такое взаимодействие GUI с традиционными конфигурационными методами Asterisk, чтобы ничто не могло воспрепятствовать его применению. Большинство GUI для Asterisk используют формат промежуточной конфигурации или базу данных, с помощью которых можно создать конфигурационные файлы для использования Asterisk. К сожалению, это означает, что любая опция, не представленная в GUI, не может быть задана «вручную» в конфигурационных файлах. А вот Asterisk GUI реально изменяет традиционные конфигурационные файлы Asterisk, то есть изменения, вносимые в GUI, и изменения, вносимые в сами файлы, могут сосуществовать и даже передаваться туда и обратно. Например, если изменить ID вызывающего абонента в файле users.conf и обновить GUI, изменения можно будет увидеть и в GUI. Аналогично, если внести изменения в GUI и перезагрузить файл, изменения

будут отражены и в нем. Если добавить новые настройки, не представленные в GUI (например, ввести `nat=yes` в конкретную запись в файле users.conf) и затем изменить ID вызывающего абонента в GUI, вы увидите, что строка `nat=yes` сохранится в файле даже несмотря на то, что произойдет изменение ID вызывающего абонента. Комментарии тоже обычно сохраняются при редактировании через GUI. Это не только означает, что GUI больше не должен отображать все возможные конфигурации, поскольку наиболее специфичные из них могут быть заданы вручную. Это также означает, что, если кто-то начнет с использования Asterisk GUI, а затем выйдет за его рамки, он вполне естественным образом сможет создавать более сложные функции, не отказываясь от уже ставшего привычным GUI.

## Использование GUI

При первой регистрации во вновь созданном GUI система активирует Мастер настройки, который позволяет настроить основные элементы системы телефонной связи.

GUI может не суметь определить все типы интерфейсов TDM и в результате сообщит, что не находит определенные платы, даже несмотря на то, что они установлены. Предполагается, что со временем GUI научится определять любые платы, использующие интерфейс Zaptel, и работать с ними, но эта функциональность обещает быть сложной и на данный момент находится на этапе разработки.

Мастер настройки одну за другой предлагает некоторые базовые опции, такие как длина добавочного номера или правила набора. Мы не собираемся углубляться в детали того, как работает стандартный GUI. Он находится в процессе непрерывной разработки, и, скорее всего, читая эту книгу, вы не найдете в GUI многое из того, что мы могли бы написать здесь.

## Элементы GUI

Стандартный GUI, который поставляется с AsteriskNOW (или который можно скачать через SVN), имеет стандартный набор элементов. Эти элементы представляют собой то, что может присутствовать в типовой малой офисной АТС. В настоящее время в меню представлены следующие элементы:

- Users (Абоненты).
- Conferencing (Конференц-связь).
- Voicemail (Голосовая почта).
- Call Queues (Очереди звонков).
- Service Providers (Поставщики сервисов).
- Calling Rules (Правила вызова).
- Incoming Calls (Входящие звонки).

- Voice Menus (Голосовые меню).
- Record a Menu (Запись в меню).
- Active Channels (Активные каналы).
- Graphs (Диаграммы).
- System Info (Сведения о системе).
- Backup (Создание резервной копии).
- Options (Опции).

# Архитектура Asterisk GUI

Прежде чем углубляться в изучение (или разработку собственного) Asterisk GUI, важно понимать, как реализуется поток данных между клиентом (веб-броузером) и Asterisk. Поскольку эти интерфейсы являются Ajax-приложениями, многое в них не вполне понятно. Поток управления проходит примерно так:

- Броузер переходит по URL вашего приложения управления.
- Веб-сервер Asterisk отправляет броузеру HTML-страницу, библиотеки и само приложение (которое написано на JavaScript и активно использует Ajax).
- Пользователь взаимодействует с броузером; по мере необходимости приложение JavaScript присылает команды назад веб-серверу. Эти команды, представленные в форме URL, запрашивают некоторое действие от самого сервера Asterisk.
- Веб-сервер интерпретирует эти URL. Если пользователь прошел регистрацию успешно, он посылает команду (действие) Asterisk через Asterisk Manager Interface (AMI), описанный в главе 10.
- Asterisk выполняет действия и передает результаты (код состояния и, возможно, данные) на веб-сервер.
- Веб-сервер возвращает ответ Asterisk JavaScript-приложению, выполняющемуся в броузере.
- JavaScript-приложение обновляет окно броузера.

Несмотря на то что на первый взгляд это может показаться несколько сложным, не пугайтесь. Это очень гибкая и мощная архитектура, которая может использоваться для разнообразнейших приложений, а не только для Asterisk GUI. Однако сейчас сосредоточимся на совершенствовании Asterisk GUI. Начнем с настройки базовых частей и затем перейдем к установке и изменению Asterisk GUI.

## Компоненты Asterisk GUI

Рассмотрим подробнее некоторые ключевые компоненты Asterisk GUI. Они будут использоваться позже в данной главе для изменения Asterisk GUI.

### Asterisk Manager Interface

Как рассказывалось в главе 10, Asterisk Manager Interface обеспечивает возможность внешним программам управлять Asterisk. Интерфейс Manager – это сердце Asterisk GUI, поскольку он выполняет всю тяжелую работу.

### Команды Manager по HTTP и веб-сервер Asterisk

Веб-сервер, встроенный в Asterisk, позволяет передавать команды интерфейса Manager в Asterisk по HTTP, а не подключаться непосредственно к интерфейсу Manager. Это намного упрощает для веб-приложения задачу по передаче команд AMI в Asterisk с использованием Asynchronous JavaScript Asterisk Manager (AJAM), что мы вскоре рассмотрим. Веб-сервер также можно конфигурировать на обслуживание статического содержимого, такого как HTML-файлы и изображения[1].

### AJAM и JavaScript

Инфраструктура AJAM использует JavaScript и XML для асинхронной отправки команд в Asterisk и обновления информации, отображаемой в веб-броузере.

## Установка Asterisk GUI

Если у вас не установлен AsteriskNOW, необходимо скачать и установить файлы Asterisk GUI. После загрузки эти файлы просто компилируются и устанавливаются как часть Asterisk.

Для использования Asterisk GUI необходима Asterisk версии 1.4 или более поздней.

Самую последнюю версию файлов GUI можно найти в хранилище Subversion компании Digium[2]. Если на вашем компьютере установлено Subversion, код GUI можно загрузить, используя следующую команду:

---

[1] Возможно, вы спрашиваете себя: «Почему веб-сервер встроен в Asterisk? Почему бы просто не использовать внешний веб-сервер?» Внешний веб-сервер может использоваться для обслуживания Asterisk GUI, но это выходит за рамки рассмотрения данной главы, поскольку модель безопасности, лежащая в основе Ajax, разрешает Ajax направлять запросы только к тому домену, порту и по тому протоколу, по которым поступила HTML-страница. Обычно такое поведение называют политикой единства происхождения.

[2] В настоящее время нет способа загрузить GUI через FTP. Эта ситуация может измениться в любой момент, поэтому не стесняйтесь и свободно проверяйте, не появилась ли обновленная информация на веб-сайте Asterisk.

```
# cd /usr/src # или любая папка, в которую вы хотите загрузить исходный код
# svn co http://svn.digium.com/svn/asterisk-gui/trunk asterisk-gui
```

Установить GUI очень просто:

```
# cd asterisk-gui
# ./configure
# make
# make install
# make samples
```

После выполнения представленных выше команд файлы GUI будут установлены и станут частью вашего дистрибутива Asterisk.

## Настройка httpd.conf и manager.conf

Конфигурация веб-сервера Asterisk для обработки запросов AJAM включает несколько простых шагов. В файл /etc/asterisk/http.conf необходимо добавить (или раскомментировать) следующее:

```
[general]
enabled=yes
enablestatic=yes ; без этого вы можете только посылать команды AMI,
                 ; но не отображать html-содержимое

bindaddr=0.0.0.0 ; адрес, на который HTTP-сервер Asterisk должен отвечать
bindport=8088    ; порт, по которому HTTP-сервер Asterisk должен отвечать
prefix=asterisk  ; будет формировать часть URI, соответствующую имени папки
```

Теперь, когда httpd.conf настроен, можно передать содержимое в браузер. Чтобы веб-клиент мог посылать команды в Asterisk, необходимо внести некоторые изменения в Asterisk Manager Interface (AMI). Для этого добавим несколько строк в раздел `[general]` файла manager.conf и учетную запись пользователя с набором разрешений `config`. Откроем файл manager.conf и отредактируем его следующим образом:

```
[general]
enabled=yes    ; возможно, AMI уже активирован, если используется для других целей
webenabled=yes ; это активирует взаимодействие между веб-сервером Asterisk и AMI

[asterisk_http] ; пользователю может быть присвоено любое имя
secret = gooey
read = system,call,log,verbose,command,agent,user,config
write = system,call,log,verbose,command,agent,user,config
```

Сохраните изменения и перезапустите Asterisk. У вас должно получиться подключиться в веб-серверу Asterisk посредством следующего URI:

```
http://localhost:8088/asterisk/static/ajamdemo.html
```

Если по какой-то причине возникли проблемы с переходом на демонстрационную страницу, вернитесь в папку исходного кода asterisk-gui и выполните команду

```
# make checkconfig
```

Вот и все! Asterisk теперь поддерживает веб-доступ. Пора переходить к реальной разработке с использованием Asterisk GUI.

# Формирование Asterisk GUI

После установки файлов для Asterisk GUI можно приступать к формированию GUI. В следующих нескольких разделах поэтапно рассматриваются настройка и объединение различных компонентов с целью улучшения и расширения возможностей GUI.

## Передача команд интерфейса Manager по HTTP

Asterisk GUI формирует команды для Asterisk, вызывая специально созданные URL на веб-сервере Asterisk. В этом разделе представлены примеры некоторых обычно используемых команд (действий) и соответствующие ответы веб-сервера. Эти URL AMI имеют следующую общую структуру:

```
http://hostname:8088/asterisk/rawman?action=команда&....пары параметр=значение...
http://hostname:8088/asterisk/manager?action=команда&....пары параметр=значение...
http://hostname:8088/asterisk/mxml?action=команда&....пары параметр=значение...
```

Разница между URL `rawman`, `manager` и `mxml` важна. Веб-сервер экспортирует три разных представления интерфейса AMI. Если используется URL `rawman`, сервер возвращает в HTTP-ответе последовательность пар ключевое слово/значение. Если используется URL `manager`, сервер возвращает результат в HTML-формате. Аналогично, если используется URL `mxml`, сервер возвращает результаты в XML-формате. Для современных приложений в стиле Ajax формы `rawman` и `mxml`, пожалуй, более полезны[1].

Действия с параметрами, которые могут быть переданы на сервер, являются обычными командами интерфейса управления, описываемыми в приложении F. Обратите внимание: действия `LOGIN` и `CHALLENGE` уникальны тем, что посылаются не непосредственно в Asterisk, а обрабатываются интерфейсом Manager для аутентификации пользователя. Если пользователь не прошел аутентификацию, сервер не передает действие на обработку в Asterisk, а возвращает ошибку.

Ознакомимся с некоторыми широко используемыми действиями и рассмотрим, как их можно использовать для управления сервером.

### LOGIN

Команда `LOGIN` аутентифицирует учетные данные для доступа к HTML-представлению интерфейса Manager. Как только вы зарегистрировались, Asterisk сохраняет в вашем броузере объект cookie (который дейст-

[1] По той же причине людям намного проще использовать для отладки форму `manager`.

вителен в течение времени, заданного настройкой `httptimeout`). Этот cookie используется для подключения к одному и тому же сеансу. URL

*http://localhost:8088/asterisk/rawman?action=login&username=asterisk_http &secret=gooey*

отправляет на веб-сервер команду на регистрацию, которая включает учетные данные. Если регистрация прошла успешно, сервер отвечает следующим образом:

```
Response: Success
Message:  Authentication accepted
```

Это, конечно, очень упрощенное представление принципа работы регистрации. Отправка имени пользователя и пароля в URL является плохой практикой, хотя и очень полезной при формировании GUI. Более подходящим способом реализации регистрации и примером более сложной обработки команды является использование последовательности запрос/ответ. Сформируйте такой запрос:

*http://localhost:8088/asterisk/rawman?action=challenge&AuthType=md5*

Команда `CHALLENGE` запускает последовательность запрос/ответ, которая может использоваться для регистрации пользователя. Сервер отвечает, отправляя запрос (произвольную строку) в ответе:

```
Response: Success
Challenge: 113543555
```

Ваше приложение отвечает на запрос, вычисляя хеш MD5 запроса, конкатенированного с паролем пользователя. Вот как пользователь может вручную вычислить хеш MD5:

```
# echo -n 113543555gooey | md5sum
          50a0f43ad4c9d99a39f1061cf7301d9a -
```

После этого вычисленный хеш может использоваться как ключ регистрации в URL:

*http://localhost:8088/asterisk/rawman?action=login&username=asterisk_http&authtype=md5&key=50a0f43ad4c9d99a39f1061cf7301d9a*

В целях безопасности регистрация должна произойти в течение пяти секунд после запроса. Также обратите внимание: чтобы система запрос/ответ работала, в броузере должен быть активирован прием объектов cookie, поскольку именно cookie гарантирует, что действие регистрации использует тот же ID сеанса интерфейса управления, что и действие запроса.

Если для запроса используется URL `manager` (а не `rawman`), ответ будет получен в формате HTML:

```
<title>Asterisk&trade; Manager Interface</title>
<body bgcolor="#ffffff">
<table align=center bgcolor="#f1f1f1" width="500">
```

```
<tr><td colspan="2" bgcolor="#f1f1ff"><h1>  Manager Tester</h1></td>
</tr>
<tr><td>Response</td><td>Success</td></tr>
<tr><td>Challenge</td><td>113543555</td></tr>
</table>
</body>
```

Аналогично, если используется представление `mxml`, будет получен ответ в формате XML:

```
<Ajax-response>
   <response type='object' id='unknown'>
      <generic response='Success' challenge='113543555' />
   </response>
</Ajax-response>
```

Кроме формата, эти три типа ответов больше ничем не отличаются. Для большинства приложений в подобной ситуации, когда нет необходимости отображать HTML-страницу пользователю, извлечь запрос из пар ключевое слово/значение будет намного проще, чем использовать `rawman` или `mxml`.

## Передача вызова

Действие `REDIRECT` может использоваться для передачи вызова. Просто сформируйте такой URL:

*http://localhost:8088/asterisk/rawman?action=redirect&channel=SIP/John-ae201e78&priority=1&exten=6001*

Этот URL передает заданный канал в другой добавочный номер и приоритет диалплана. Ответ на это действие такой:

```
Response: Success
Message: Redirect Successful
```

## Чтение конфигурационного файла

Команда `GETCONFIG` возвращает содержимое конфигурационного файла или его часть. HTTP-запрос

*http://localhost:8088/asterisk/rawman?action=getconfig&filename=users.conf*

возвращает содержимое файла users.conf. Asterisk GUI использует эту функциональность для представления текущей конфигурации Asterisk конечному пользователю. Ответ выглядит следующим образом:

```
Response: Success
   Category-000000: general
   Line-000000-000000: fullname=New User
   Line-000000-000001: userbase=6000
   Line-000000-000002: hasvoicemail=yes
   Line-000000-000003: hassip=yes
   Line-000000-000004: hasiax=yes
   Line-000000-000005: hasmanager=no
```

```
Line-000000-000006: callwaiting=yes
Line-000000-000007: threewaycalling=yes
Line-000000-000008: callwaitingcallerid=yes
Line-000000-000009: transfer=yes
Line-000000-000010: canpark=yes
Line-000000-000011: cancallforward=yes
Line-000000-000012: callreturn=yes
Line-000000-000013: callgroup=1
Line-000000-000014: pickupgroup=1
Line-000000-000015: host=dynamic
Category-000001: 6007
Line-000001-000000: fullname=Bill Savage
Line-000001-000001: secret=1234
Line-000001-000002: email=bsavage@digium.com
Line-000001-000003: cid_number=6001
Line-000001-000004: zapchan=
Line-000001-000005: context=numberplan-custom-1
Line-000001-000006: hasvoicemail=yes
Line-000001-000007: hasdirectory=no
Line-000001-000008: hassip=yes
Line-000001-000009: hasiax=yes
Line-000001-000010: hasmanager=no
Line-000001-000011: callwaiting=yes
Line-000001-000012: threewaycalling=yes
Line-000001-000013: mailbox=6007
Line-000001-000014: hasagent=yes
Line-000001-000015: group=
```

## Обновление конфигурационных файлов с помощью UPDATECONFIG

Действие `UPDATECONFIG` используется для обновления одной или более настроек конфигурационного файла. Например, чтобы удалить пользователя, необходимо выполнить такой HTTP-запрос:

*http://localhost:8088/asterisk/rawman?action=updateconfig&reload=yes&srcfilename=users.conf&dstfilename=users.conf&Action-000000=delcat&Cat-000000=6003& Var-000000=&Value-000000=*

## Ответ, свидетельствующий об ошибке

Чтобы формировать все остальные команды, пользователь должен зарегистрироваться на веб-сервере. Если пользователь не аутентифицирован, любая из обсуждаемых выше команд будет возвращать ошибку. Если он отправлен пользователем, не прошедшим аутентификацию, URI *http://localhost:8088/asterisk/rawman?action=ping* возвращает такой свидетельствующий об ошибке ответ:

```
Response: Error
Message:  Authentication Required
```

# Ajax, AJAM и Asterisk

Аббревиатура Ajax расшифровывается как *Asynchronous JavaScript and XML* (Асинхронный JavaScript и XML). Хотя этот термин включает слова «асинхронный» и «XML», это не означает ни то, что можно делать только асинхронные запросы, ни то, что должен обязательно использоваться XML. Некоторые авторы описывают Ajax просто как комбинацию HTML, JavaScript, DHTML и DOM. Следующее поколение броузеров, таких как Mozilla/Firefox, для отправки асинхронного запроса на сервер используют `XMLHttpRequest` (объект JavaScript). Запрос выполняется в фоновом режиме и обрабатывается сервером.

Назад в броузер результат передается посредством обратного вызова: все, что возвращает сервер, может сохраняться и использоваться для обновления отображаемой страницы. Для Internet Explorer 5 или более поздних версий той же цели служит объект ActiveX `XMLHttp`.

## Обработка форм в традиционном веб-приложении

HTML-формы обычно передаются посредством кнопки SUBMIT (ПЕРЕДАТЬ) (`type=submit`). После щелчка пользователем по кнопке SUBMIT (ПЕРЕДАТЬ) обработка веб-приложения останавливается и не возобновляется до тех пор, пока сервер не возвратит новую страницу полностью:

```
<FORM action="login.php" method="POST">
   <input type=text name="username">
   <input type=password name="password">
   <input type=submit>
</FORM>
```

Прежде чем переходить к Ajax или JavaScript, давайте рассмотрим, как работает традиционное веб-приложение. Для описания формы, где определяются все параметры, которые пользователь хочет отправить на сервер, традиционные веб-приложения используют элемент `<FORM>`. Кроме того, атрибут `action="login.php"` информирует броузер, куда отправлять все эти переменные. `method="POST"` указывает броузеру, как отправлять эти переменные на сервер.

## Обработка форм в приложении Ajax

Приложение Ajax для отправки содержимого формы на сервер использует JavaScript. Если выполнен асинхронный запрос, JavaScript-код не ожидает ответа сервера. Это также означает, что пользователи могут продолжать работать со страницей даже в том случае, если в фоновом режиме выполняется запрос. Это может представлять опасность, из-за чего, возможно, до завершения запроса потребуется ограничить некоторые действия. Броузер по умолчанию не обеспечивает визуальной индикации выполнения запроса в фоновом режиме. За информирование пользователя о выполнении запроса отвечаете вы. Вот код для передачи содержимого полей имени пользователя (`username`) и пароля (`password`) через Ajax:

```
<script language="javascript" type="text/javascript">
 function submitform(){
  var uname = document.getElementById("username").value;
  var pwd = document.getElementById("password").value;
  // xmlHttp = new ActiveXObject("Msxml2.XMLHTTP"); // IE 7
  // xmlHttp = new ActiveXObject("Microsoft.XMLHTTP"); // IE 5
  xmlHttp = new XMLHttpRequest(); // Mozilla или Firefox

  var url = "/rawman?action=login&username=" + escape(uname) + "&secret=" +
  escape(pwd);

  xmlHttp.open("GET", url, true);
  xmlHttp.onreadystatechange = dosomething;
  // вместо dosomething использовалась бы функция JavaScript
  xmlHttp.send(null);
}
</script>
```

Метод `getElementById()` читает значения полей имени пользователя и пароля. Затем создается объект `XMLHttpRequest`, с помощью которого эти значения отправляются назад на сервер. Заметьте, что тип используемого объекта зависит от того, какой броузер применяют пользователи: Internet Explorer 7, 5 или Mozilla/Firefox. Довольно просто написать код, обрабатывающий все эти ситуации, или использовать библиотеку, подобную Prototype, для реализации независимости от платформы. Имя пользователя и пароль кодируются в URL и отправляются на сервер. Вызов `xmlHttp.onreadystatechange` регистрирует обработчик для обработки результата, возвращаемого сервером.

Данный код реализует только запрос `XMLHttp` и указывает броузеру вызвать функцию `dosomething()` при получении ответа от сервера. Вот функция `dosomething()`, которая обрабатывает этот ответ:

```
<script language="javascript" type="text/javascript">
   function dosomething() {
      if (xmlHttp.readyState == 4) {
         var login_response = xmlHttp.responseText;
      }
   }
</script>
```

Перед выполнением каждого последующего этапа `XMLHttp`-запроса убедитесь в завершении предыдущего (успешном или со сбоем).

Эта функция вызывается при любом изменении состояния HTTP-запроса. Выражение `if` сохраняет ответ, только если параметр `readyState` запроса имеет значение 4, свидетельствующее о завершении запроса. Теперь переменная JavaScript `login_response` (ответ на регистрацию) содержит ответ страницы регистрации.

Но этот фрагмент мало похож на код, готовый к производственной эксплуатации. В частности, упрощенная обработка имени пользователя и пароля подходит для тестирования, но будет представлять серьезную проблему для безопасности в производственной системе, даже если приложение используется только в закрытой сети. Для реализации более надежной и безопасной обработки паролей рекомендуется применять представленную ранее систему запрос/ответ. Больше информации о написании веб-приложений на Ajax предлагает книга Бретта Маклафлина (Brett McLaughlin) «Head Rush Ajax» (издательство O'Reilly).

## Инфраструктура Prototype

Prototype (*http://prototypejs.org*) – это инфраструктура JavaScript, выпущенная под лицензией типа MIT. Prototype может чрезвычайно упростить разработку Ajax-приложения. Она предоставляет много способов сделать код более коротким и ясным. Например, в функции `submitform` вызов `document.getElementById()` может быть заменен функцией `$()`. Аналогично, вызов `value` для получения содержимого DOM-элемента можно заменить вызовом `$F()`. Таким образом, `document.getElementById("username").value` становится просто `$F('username')`; в результате получаем намного более простой и удобный для чтения код.

Prototype также позволяет выполнять элегантные `XMLHttp`-запросы. С помощью объекта Prototype `Ajax` функцию `submitform()` можно переписать следующим образом:

```
<script language="javascript" type="text/javascript">
function submitform(){
   var url = '/rawman';
   var pars = 'username=' + escape($F('username')) + '&secret=' + escape
   ($F('password'));

var myAjax = new Ajax.Request( url,
   { method: 'get',
     parameters: pars,
     onComplete: dosomething
   });

}
</script>
```

Этот код не только намного короче, теперь в веб-страницы не придется включать специальный код для каждого броузера. Prototype берет на себя заботу о различиях между Mozilla/Firefox и версиями Internet Explorer. Более того, она выполняет проверку параметра `readyState` запроса, поэтому больше нет необходимости в этом ужасном выражении `if`. Prototype имеет массу встроенных функций, некоторые из них активно используются в инфраструктуре Asterisk. Обсуждать их здесь нет возможности, но более подробную информацию можно найти в разделе «Short Cuts» книги «Prototype Quick Reference» Скотта Реймонда

(Scott Raymond) и «Prototype and Scriptaculous: Taking the Pain Out of JavaScript» Криса Ангуса (Chris Angus), обе книги изданы в O'Reilly.

## Настройка GUI

Рассмотрев разные части, формирующие основу Asterisk GUI, мы располагаем всем необходимым для исследования самого GUI и его настройки соответственно собственным нуждам. Asterisk GUI можно найти по следующему адресу: *http://localhost:8088/asterisk/static/config/cfgbasic.html*.

Взглянув на рис. 11.1, вы можете подумать, что Asterisk GUI – это просто еще один из множества существующих графических пользовательских интерфейсов Asterisk. Но это абсолютно не так. Этот GUI не просто позволяет настраивать себя, он практически умоляет сделать это. В данном разделе обсуждается, как можно настраивать GUI и использовать AJAM для создания собственных расширений к GUI. Чтобы извлечь из этой информации максимальную пользу, необходимо обладать базовыми знаниями по HTML и JavaScript.

Домашняя страница GUI называется cfgbasic.html. Все остальные страницы загружаются в окно iframe, содержащееся на странице cfgbasic.html. По умолчанию cfgbasic.html загружает home.html в основном окне.

Большинство изменений GUI в конечном счете связаны с изменением файла cfgbasic.html, который формирует окно регистрации.

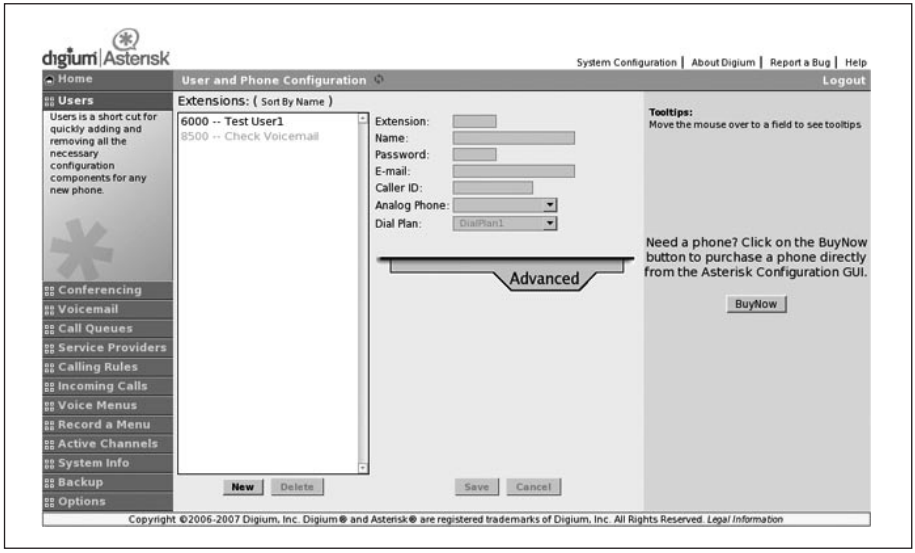



***Рис. 11.1.*** *Скриншот Asterisk GUI*

## Как добавить в GUI новую вкладку

В качестве примера настройки Asterisk GUI создадим новую вкладку, отображающую содержимое файла extensions.conf. Сначала необходимо создать файл и поместить его в папку /var/lib/asterisk/static-html/config. В данном примере назовем файл test.html:

```
<script src="scripts/prototype.js"></script>
<script src="scripts/astman.js"></script>
<script>
function localAjaxinit() {
      parent.loadscreen(this);
      makerequest('g','extensions.conf', '' , function(t){
         $('ExtensionsDotConf').innerHTML = "<PRE>" + t + "</PRE>";
      });
}
</script>
<body onload="localAjaxinit()" bgcolor="EFEFEF">
      <div id="ExtensionsDotConf"></div>
</body>
```

Данный код просто отображает конфигурацию файла extensions.conf. Конечно, это очень простой пример, но он демонстрирует основы создания новой страницы для Asterisk GUI. Рассмотрим этот пример шаг за шагом.

Первая строка указывает броузеру загрузить библиотеку Prototype. Вторая – загрузить файл astman.js, в котором содержится бóльшая часть кода для взаимодействия с интерфейсом Manager.

Далее описываем функцию `localAjaxinit`. Сначала функция `localAjaxinit` указывает родителю этой страницы (в данном случае файлу cfgbasic.html) выполнить функцию `loadscreen`, передавая в нее эту страницу как параметр. Таким образом, основное окно GUI загрузит нашу новую страницу test.html в iframe. Следующее, что мы делаем в функции `localAjaxinit`, – используем функцию `makerequest`. Функция определена в файле astman.js и очень упрощает выполнение запросов к веб-серверу[1].

Первый параметр функции `makerequest` определяет тип выполняемого запроса. Он может принимать следующие значения:

`'g'`

Использует действие `GetConfig` для извлечения конфигурации из конфигурационного файла, заданного во втором параметре.

`'u'`

Использует действие `UpdateConfig` для обновления конфигурации в конфигурационном файле, заданном во втором параметре. Третий

[1] Фактически `makerequest` – это просто оболочка вызова Prototype-метода `Ajax.Request`.

параметр функции определяет данные конфигурации, которые должны быть обновлены.

''

Если в качестве первого параметра функции `makerequest` заданы одинарные кавычки, будет отправлено специальное действие, определенное в третьем параметре.

Четвертый параметр – это функция обратного вызова, которая будет вызываться в ответ на запрос Ajax.

### Примеры использования makerequest

Возьмем в качестве примера следующий фрагмент кода. Он демонстрирует три разных способа использования функции `makerequest`. В первом случае мы получаем конфигурационные данные из файла users.conf. Во втором – обновляем musiconhold.conf и меняем значение настройки `random` в классе `default`. И последнее (по порядку, но не по значимости) – вызываем действие `Ping`. В каждом случае задается функция обратного вызова `t`, которая просто заменяет содержимое переменной `div` ответом, полученным в результате Ajax-вызова.

```
makerequest( 'g', 'users.conf', '' ,
   function(t) { $('ExtensionsDotConf').innerHTML = "<PRE>" + t +
   "</PRE>"; } );
makerequest( 'u', 'musiconhold.conf',

'&Action-000000=update&Cat-000000=default&Var-000000=random&Value-
   000000=yes' ,
   function(t) { $('ExtensionsDotConf').innerHTML = "<PRE>" + t +
   "</PRE>"; } );

makerequest( '', '', 'action=Ping' ,
   function(t) { $('ExtensionsDotConf').innerHTML = "<PRE>" + t +
   "</PRE>"; } );
```

Все остальное содержимое test.html – это просто HTML-код с элементом `div`, в котором будут размещены конфигурационные данные после их получения. Обратите внимание, что тег HTML-кода имеет атрибут `onload`, который обусловливает выполнение броузером функции `localAjaxinit` сразу же по завершении загрузки страницы.

Теперь, когда новая страница создана, необходимо отредактировать файл cfgbasic.html, чтобы добавить эту страницу как панель GUI. Откройте файл cfgbasic.html, найдите JavaScript-функцию `returnpanels` и вставьте этот код в список панелей в том месте, где вы хотите разместить свою панель:

```
newpanel( ["Test", "test.html", "Test"]);
```

Теперь перезагрузите GUI в своем броузере. Слева должна появиться новая вкладка Test (Проверка), и после щелчка по ней будут отображаться значения конфигурации для файла extensions.conf.

Это всего лишь малая часть того, что можно рассказать об интерфейсе AJAM и Asterisk GUI, но данный пример служит только для того, чтобы продемонстрировать вам, как просто вводить новую функциональность в GUI. В следующем примере будет показано, как легко выводить в GUI настройки из конфигурационных файлов.

## Вывод настроек конфигурации в GUI

Как говорилось ранее, одно из уникальных преимуществ Asterisk GUI по сравнению с другими графическими интерфейсами для Asterisk в том, что он обновляет существующие конфигурационные файлы, предпринимая при этом специальные меры для предотвращения перезаписи или удаления каких-либо дополнительных настроек, которые могут присутствовать в конфигурационных файлах. Чтобы продемонстрировать ту простоту, с которой можно предоставлять новые настройки в GUI, добавим в GUI простой флажок, обеспечивающий возможность задавать настройку `nat` в файле users.conf.

Если открыть GUI и щелкнуть по вкладке Users (Пользователи), GUI загрузит файл users.html в iframe. Откроем afqk users.html (обычно располагающийся в gfgrt /var/lib/asterisk/static-http/config) и начнем изменять его, чтобы добавить наш флажок.

Сначала посмотрим в начало файла, где описана переменная `fieldnames` (имена полей). Эта переменная содержит список всех имен полей, которые будут заданы на данной странице GUI. Просто добавьте `nat` в конец списка или вставьте следующую строку прямо под текущим описанием `fieldnames`.

```
fieldnames.push('nat');
```

Таким образом мы сообщаем Asterisk GUI о том, что хотим иметь возможность видеть, а также задавать значение `nat`. Однако, чтобы увидеть или задать значение, необходимо добавить элемент в HTML-форму. Для этого найдите в файле users.html флажок IAX и добавьте следующие строки между ним и флажком CTI.

```
<tr>
   <td align=right><input type='checkbox' id='nat'></td>
   <td>NAT</td>
</tr>
```

Просто перезагрузите страницу – вот и все. Всего несколько дополнительных строк кода – и мы можем выводить настройку `nat` в GUI. Проще быть не может!

Занимаясь формированием Asterisk GUI, вы, вероятно, обнаружите, что отладка кода Ajax и JavaScript порой может вызывать некоторые трудности. Мы настоятельно рекомендуем использовать расширение для Mozilla/Firefox под названием Firebug, которое существенно упрощает задачу по отладке Ajax, JavaScript и HTML. Его можно найти по адресу *http://www.getfirebug.com*. Существует также упрощенная версия для Internet Explorer, известная как Firebug Lite, которую можно скачать на том же веб-сайте.

## Дополнительная информация

В данной главе были представлены Asterisk GUI и инфраструктура AJAM. Мы рассмотрели модель работы GUI и то, как его можно изменять. Дополнительную информацию по разработке графического интерфейса для Asterisk можно найти в руководстве для разработчиков GUI (GUI Developers Guide) по адресу *http://asterisknow.org/developers/gui-guide*.

# 12

# Интеграция с реляционной базой данных

*Ничто так не раздражает, как хороший пример.*
– Марк Твен

## Введение

В данной главе мы собираемся исследовать вопросы интеграции некоторых функций Asterisk и системы управления базами данных (СУБД). Для работы с Linux доступно несколько СУБД, но мы решили ограничить наше обсуждение PostgreSQL. Мы прекрасно понимаем, что MySQL – тоже чрезвычайно популярная СУБД, но надо было остановиться на какой-то одной, и наш опыт работы с PostgreSQL перевесил чашу весов в ее пользу. Фактически будет обсуждаться ODBC-коннектор, поэтому, если ваша база данных поддерживает работу через ODBC, содержание данной главы будет вам полезным.

Интеграция Asterisk с базами данных – один из основополагающих элементов, обеспечивающих возможность включения Asterisk в большую распределенную систему. Используя мощь базы данных, динамически меняющиеся данные могут переносить информацию по массиву систем Asterisk. Наш последний фаворит среди функций Asterisk – `func_odbc`, которая будет рассмотрена в данной главе позже.

Не только развернутые системы Asterisk используют реляционные базы данных; понимание того, как работать с ними, открывает сундук с сокровищами, полный новых способов разработки решений для сетей связи.

# Установка СУБД PostgreSQL

Первое, что надо сделать, – это установить сервер базы данных PostgreSQL[1].

```
# yum install -y postgresql-server
```

Затем запускаем базу данных; первая инициализация займет несколько секунд:

```
# service postgresql start
```

Далее создаем пользователя asterisk, под учетной записью которого будут выполняться соединение и управление базой данных. Выполним следующие команды:

```
# su - postgres
$ createuser -P
Enter name of user to add: asterisk
Enter password for new user:
Enter it again:
Shall the new role be a superuser? (y/n) n
Shall the new user be allowed to create databases? (y/n) y
Shall the new user be allowed to create more new users? (y/n) n
CREATE USER
```

По умолчанию PostgreSQL не слушает TCP/IP-соединение, которое будет использовать Asterisk. Необходимо внести изменения в файл /var/lib/pgsql/data/postgresql.conf, чтобы Asterisk могла устанавливать IP-соединения с базой данных. Для этого просто удалим символ комментария перед параметрами `tcpip_socket` и `port`. Не забудьте изменить значение `tcpip_socket` с `false` на `true`.

```
tcpip_socket = true max_connections = 100
      # примечание: увеличение max_connections стоит
      # выделения примерно 500 байтов совместно
      # используемой памяти на каждое соединение,
      # кроме памяти, выделяемой под shared_buffers
      # и max_locks_per_transaction.
#superuser_reserved_connections = 2
port = 5432
```

Теперь редактируем файл /var/lib/pgsql/data/pg_hba.conf, чтобы обеспечить возможность только что созданному пользователю asterisk устанавливать соединения с сервером PostgreSQL через TCP/IP. В конце файла замените все под комментарием `# Разместите здесь свою фактическую конфигурацию` следующим[2]:

```
host   all  asterisk  127.0.0.1  255.255.255.255   md5
local  all  asterisk                               trust
```

---

[1] Для большой, сильно загруженной системы рекомендуется устанавливать его отдельно от системы Asterisk, на другом компьютере.

[2] В данном примере серверу Asterisk разрешается устанавливать соединение с PostgreSQL и запрашивать пароль на доступ. – *Примеч. науч. ред.*

Теперь можно создать базу данных, которая будет использоваться в данной главе. Мы собираемся создать базу данных `asterisk` и задать в качестве владельца нашего пользователя asterisk.

```
$ createdb --owner=asterisk asterisk
CREATE DATABASE
```

Выйдя из учетной записи postgres и вернувшись в административную учетную запись, перезапустите сервер PostgreSQL:

```
$ exit
# service postgresql restart
```

Наше соединение с сервером PostgreSQL через TCP/IP можно проверить следующим образом:

```
# psql -h 127.0.0.1 -U asterisk Password:
Welcome to psql 7.4.16, the PostgreSQL interactive terminal.

Type: \copyright for distribution terms
      \h for help with SQL commands
      \? for help on internal slash commands
      \g or terminate with semicolon to execute query
      \q to quit

asterisk=>
```

Повторно проверьте свою конфигурацию, как было показано ранее, если получена следующая ошибка, которая свидетельствует о том, что соединение через TCP/IP не разрешено:

```
psql: could not connect to server: Connection refused
Is the server running on host "127.0.0.1" and accepting TCP/IP connections on
port 5432?
```

(psql: невозможно установить соединение с сервером: В соединении отказано

Сервер работает на хосте “127.0.0.1” и принимает TCP/IP-соединения через порт 5432?)

# Установка и конфигурация ODBC

ODBC-коннектор – это уровень обобщения баз данных, который делает возможным взаимодействие Asterisk с разнообразными СУБД без необходимости создания специального коннектора для каждой отдельно взятой базы данных, которую должна поддерживать Asterisk. Это устраняет необходимость в затратах усилий на разработку и обслуживание кода. Имеется некоторое негативное влияние на производительность, потому что между Asterisk и базой данных вводится дополнительный уровень приложений, однако его можно смягчить надлежащим проектированием, и это стоит выполнить, если необходимо обеспечить мощную и гибкую функциональность баз данных в системе Asterisk.

Прежде чем устанавливать коннектор в Asterisk, необходимо установить ODBC в самой Linux. Чтобы установить драйверы ODBC, просто выполним следующую команду:

```
# yum install -y unixODBC unixODBC-devel libtool-ltdl libtool-ltdl-devel
```

В главе 3 можно найти таблицу с пакетами, которые должны быть установлены.

Необходимо установить пакет unixODBC-devel, потому что Asterisk использует его для создания модулей ODBC, которые будут применяться в данной главе.

Убедитесь в наличии сконфигурированного ODBC-драйвера PostgreSQL в файле /etc/odbcinst.ini. Он должен выглядеть примерно так:

```
[PostgreSQL]
Description    = ODBC for PostgreSQL
Driver         = /usr/lib/libodbcpsql.so
Setup          = /usr/lib/libodbcpsqlS.so
FileUsage      = 1
```

Убедитесь, что система видит драйвер, выполнив приведенную ниже команду. Если все хорошо, она должна возвратить имя метки используемой СУБД `PostgreSQL`.

```
# odbcinst -q -d
[PostgreSQL]
```

Далее сконфигурируем файл /etc/odbc.ini, используемый для создания коннектора, который Asterisk будет применять для ссылки на эту конфигурацию. Если когда-то в будущем понадобится изменить базу данных или что-то еще, просто надо будет внести изменения в этот файл, не меняя ссылок в Asterisk[1].

```
[asterisk-connector]
Description       = PostgreSQL connection to 'asterisk' database
Driver            = PostgreSQL
Database          = asterisk
Servername        = localhost
UserName          = asterisk
Password          = welcome
Port              = 5432
Protocol          = 7.4
ReadOnly          = No
RowVersioning     = No
ShowSystemTables  = No
ShowOidColumn     = No
FakeOidIndex      = No
ConnSettings      =
```

Убедимся, что мы можем соединяться с нашей базой данных, используя приложение `isql`. Приложение `isql` не будет осуществлять соединение как пользователь с правами администратора (root) и должно выпол-

[1] Да, слишком много всего. На самом деле нужны только записи `Driver`, `Database` и `Servername`. Даже `Username` и `Password` задаются в другом месте, как вы увидите позже.

няться под учетной записью владельца базы данных. Поскольку владельцем базы данных asterisk в PostgreSQL является пользователь asterisk, необходимо создать учетную запись Linux с таким же именем. В главе 14 эта учетная запись будет использоваться для запуска Asterisk от лица пользователя, не имеющего прав администратора.

```
# su - asterisk
$ echo "select 1" | isql -v asterisk-connector
+---------------------------------------+
| Connected!                            |
|                                       |
| sql-statement                         |
| help [tablename]                      |
| quit                                  |
|                                       |
+---------------------------------------+
SQL> +------------+
| ?column?   |
+------------+
| 1          |
+------------+
SQLRowCount returns 1
1 rows fetched
$ exit
```

После установки, конфигурации и проверки unixODBC, необходимо повторно скомпилировать Asterisk, чтобы модули ODBC были созданы и установлены. Вернитесь в папку исходного кода Asterisk и запустите сценарий ./configure, чтобы система знала, что unixODBC установлен.

```
# cd /usr/src/asterisk-1.4
# make distclean
# ./configure
# make menuselect
# make install
```

Практически все, о чем говорится в данной главе, включено по умолчанию. Выполнив команду make menuselect, убедитесь, что модули, связанные с ODBC, активированы. Сюда относятся cdr_odbc, func_odbc, func_realtime, pbx_realtime, res_config_odbc, res_odbc. Для хранения голосовой почты в базе данных, совместимой с ODBC, не забудьте выбрать пункт ODBC STORAGE (ХРАНИЛИЩЕ ODBC) в меню Voicemail Build Options (Опции сборки голосовой почты). Чтобы убедиться в существовании модулей, загляните в папку /usr/lib/asterisk/modules/.

## Конфигурация res_odbc для обеспечения доступа к базе данных

Конфигурация ODBC-коннекторов выполняется в файле res_odbc.conf, располагающемся в папке /etc/asterisk. Файл res_odbc.conf задает па-

раметры, которые будут использоваться различными модулями Asterisk для соединения с базой данных[1].

Внесем изменения в файл res_odbc.conf:

```
[asterisk]
enabled => yes
dsn => asterisk-connector
username => asterisk
password => welcome
pooling => no
limit => 0
pre-connect => yes
```

Опция `dsn` указывает на соединение с базой данных, которое мы конфигурировали в файле /etc/odbc.ini, а опция `pre-connect` говорит Asterisk открыть и обслуживать соединение с базой данных при загрузке модуля res_odbc.so. Это снижает некоторые издержки, которые возникли бы при многократном повторном установлении и разрыве соединения с базой данных.

Сконфигурировав res_odbc.conf, запустите Asterisk и проверьте соединение с базой данных с помощью CLI-команды `odbc show`:

```
*CLI> odbc show
Name: asterisk
DSN: asterisk-connector
Pooled: no
Connected: yes
```

# Использование архитектуры реального времени

Архитектура реального времени Asterisk (Asterisk Realtime Architecture, ARA) – это метод хранения конфигурационных файлов (которые при обычных обстоятельствах располагались бы в папке /etc/asterisk) и их конфигурационных опций в таблице базы данных. Существует два типа архитектур реального времени: *статическая* и *динамическая*. Статический вариант аналогичен традиционному методу чтения конфигурационного файла, за исключением того что чтение данных осуществляется из базы данных. Метод динамической реализации архитектуры реального времени используется для таких элементов, как объекты `user` и `peer` (SIP, IAX2) и голосовая почта, которые загружают и обновляют информацию по мере необходимости. Изменение в статической информации требует перезагрузки текстового файла сразу после внесения в него изменений, но динамическая информация запрашивается Asterisk по мере необходимости и не требует перезагрузки. Архитекту-

[1] Опции `pooling` (создание пула) и `limit` (предел) довольно полезны для работы с базами данных MS SQL Server и Sybase. Они позволяют устанавливать с базой данных множество соединений (вплоть до `limit`), гарантируя при этом, что одновременно для каждого соединения выполняется только одно выражение (это обусловлено ограничением в протоколе, используемом этими серверами баз данных).

ра реального времени настраивается в файле extconfig.conf, располагающемся в папке /etc/asterisk. Этот файл указывает Asterisk, что брать из базы данных и откуда именно, что обеспечивает возможность загружать некоторые файлы из базы данных, а другие – из стандатных конфигурационных файлов.

## Статическая архитектура реального времени

Статическая архитектура реального времени используется, если требуется загрузить конфигурацию, которая обычно размещается в конфигурационных файлах в папке /etc/asterisk, из базы данных. Те же правила, которые применяются к плоским[1] файлам в системе, действуют и при использовании статической архитектуры реального времени. Например, требование выполнить команду перезагрузки из интерфейса командной строки Asterisk или перезагрузить модуль, связанный с конфигурационным файлом (то есть выполнить команду `module reload chan_sip.so`).

При использовании статической архитектуры реального времени мы указываем Asterisk, какие файлы хотим загрузить из базы данных, используя в файле extconfig.conf следующий синтаксис:

```
; /etc/asterisk/extconfig.conf
filename.conf => драйвер,базаданных[,таблица]
```

Если имя таблицы не задано, Asterisk будет использовать имя файла.

### Использование директивы preload

Большинство файлов можно загружать посредством статической архитектуры реального времени, но несколько файлов не могут быть загружены при использовании этого метода. Это файлы asterisk.conf, extconfig.conf и logger.conf. Кроме того, файлы manager.conf, cdr.conf и rtp.conf не могут загружаться из статической архитектуры реального времени, если драйверы подключения к базе данных не были загружены до запуска основного ядра Asterisk (потому что архитектура реального времени должна загрузить конфигурационные файлы до того, как модуль начнет читать свою конфигурацию). Поскольку в данной главе используется ODBC, в modules.conf пришлось бы добавить следующие строки:

```
; /etc/asterisk/modules.conf
preload => res_odbc.so
preload => res_config_odbc.so
```

[1] Плоскими являются двоичные файлы вида ключ–значение. Данные файлы позволяют быстрее осуществлять операции редактирования, добавления и удаления записей благодаря встроенным функциям Asterisk. – *Примеч. науч. ред.*

Модуль статической архитектуры реального времени читает настройки статических файлов из базы данных, используя особым образом форматированную таблицу. В PostgreSQL таблица для статической архитектуры реального времени создается с помощью следующего запроса:

```
CREATE TABLE ast_config
(
   id serial NOT NULL,
   cat_metric int4 NOT NULL DEFAULT 0,
   var_metric int4 NOT NULL DEFAULT 0,
   filename varchar(128) NOT NULL DEFAULT ''::character varying,
   category varchar(128) NOT NULL DEFAULT 'default'::character varying,
   var_name varchar(128) NOT NULL DEFAULT ''::character varying,
   var_val varchar(128) NOT NULL DEFAULT ''::character varying,
   commented int2 NOT NULL DEFAULT 0,
   CONSTRAINT ast_config_id_pk PRIMARY KEY (id)
)
WITHOUT OIDS;
```

Чтобы понимать, как Asterisk применяет строки базы данных для конфигурации различных загружаемых модулей, дадим краткое описание каждого из столбцов:

`cat_metric`

Значимость категории в рамках файла. Чем ниже значение, тем выше в файле располагается категория (см. врезку «Несколько слов о значениях»).

`var_metric`

Значимость элемента в рамках категории. Чем ниже значение, тем выше в списке располагается элемент. Применяется, например, для определения порядка расположения кодеков в файле sip.conf или iax.conf, когда требуется, чтобы `disallow=all` шел первым (значение 0), за ним – `allow=ulaw` (значение 1), а далее – `allow=gsm` (значение 2) (см. врезку «Несколько слов о значениях»).

`filename`

Имя файла, которое модуль в обычных условиях читал бы с жесткого диска системы (то есть musiconhold.conf, sip.conf, iax.conf и т. д.).

`category`

Имя раздела файла, такое как `[general]`, но не надо сохранять его в базу данных в квадратных скобках.

`var_name`

Опция, располагающаяся слева от знака равенства (то есть `disallow` – это `var_name` в выражении `disallow=all`).

`var_val`

Значение опции, располагающееся справа от знака равенства (то есть `all` – это `var_val` в выражении `disallow=all`).

commented

Любое отличное от 0 значение будет рассматриваться так, как если бы перед ним в плоском файле стояла точка с запятой (то есть оно было бы закоммментировано).

### Несколько слов о значениях

Значения в статической архитектуре реального времени используются для управления порядком, в котором объекты читаются в память. cat_metric и var_metric можно рассматривать как исходные номера строк в плоском файле. Больший cat_metric обрабатывается первым, потому что Asterisk выполняет сопоставление категорий снизу вверх (вот почему порядок пользователей и равноправных участников может иметь значение в файле sip.conf или iax.conf). Меньший var_metric в рамках категории обрабатывается первым, потому что Asterisk будет обрабатывать порядок опций в категории сверху вниз (например, чтобы обеспечить обработку disallow=all первым, для него в рамках категории должно быть задано меньшее значение, чем для allow).

Возьмем простой файл, который можно загрузить из статической архитектуры реального времени, musiconhold.conf file. Начнем с его перемещения во временную папку:

```
# cd /etc/asterisk
# mv musiconhold.conf musiconhold.conf.old
```

Чтобы удалить классы из памяти, необходимо перезапустить Asterisk. После этого, выполнив команду moh show classes, можно убедиться в отсутствии классов:

```
*CLI> restart now
*CLI> moh show classes
*CLI>
```

Итак, давайте вернем класс [default] в Asterisk, но теперь загрузим его из базы данных. Установим соединение с PostgreSQL и выполним следующие запросы INSERT:

```
INSERT INTO ast_config (filename,category,var_name,var_val)
VALUES ('musiconhold.conf','general','mode','files');
INSERT INTO ast_config (filename,category,var_name,var_val)
VALUES ('musiconhold.conf','general','directory','/var/lib/asterisk/moh');
```

Убедиться в том, что значения внесены в базу данных, можно, выполнив запрос SELECT:

```
asterisk=# select filename,category,var_name,var_val from ast_config;

filename         | category | var_name | var_val
-----------------+--------- +----------+--------
musiconhold.conf | general  | mode     | files
musiconhold.conf | general  | directory| /var/lib/asterisk/moh
(2 rows)
```

И теперь, чтобы указать Asterisk, что необходимо брать данные для musiconhold.conf из базы данных, осталось внести в файл extconfig.conf, находящийся в папке /etc/asterisk, всего одно изменение. Добавим следующую строку в конец файла extconfig.conf и сохраним его:

```
musiconhold.conf => odbc,asterisk,ast_config
```

Подключимся к консоли Asterisk и выполним перезагрузку:

```
*CLI> module reload
```

Теперь, выполнив команду `moh show classes`, можно убедиться, что наши классы для воспроизведения музыки во время ожидания загружаются из базы данных:

```
*CLI> moh show classes
Class: general
        Mode: files
        Directory: /var/lib/asterisk/moh
```

И вот, пожалуйста; musiconhold.conf загружается из базы данных. Точно так же можно организовать загрузку из базы данных других плоских файлов!

## Динамическая архитектура реального времени

Динамическая система реального времени используется для загрузки часто изменяющихся объектов: пользователей и равноправных участников SIP/IAX2, очередей и их членов и сообщений голосовой почты. Поскольку эта информация в системе может или меняться, или регулярно дополняться новыми записями, использование мощи базы данных позволит нам загружать ее по мере необходимости.

Все настройки архитектуры реального времени описываются в файле /etc/asterisk/extconfig.conf, но динамическая архитектура реального времени имеет строго определенные конфигурационные имена, такие как `sippeers`. Описание равноправного участника SIP (SIP peer) выполняется в следующем формате:

```
; extconfig.conf
sippeers => драйвер,базаданных[,таблица]
```

Имя таблицы является необязательным параметром. Если он не задан, Asterisk будет использовать предопределенное имя (то есть `sippeers`) как имя таблицы для поиска данных. В нашем примере для хранения информации равноправных участников SIP будет использоваться таблица `ast_sip peers`.

Помните, что у нас имеются и равноправные участники SIP (SIP peer), и пользователи SIP (SIP user); `peer` – это конечные точки, которым мы отправляем вызовы, а `user` направляют вызовы нам. `friend` – это сокращенная запись, определяющая оба типа конечных точек.

Таким образом, чтобы сконфигурировать Asterisk на загрузку всех равноправных SIP-участников из базы данных в режиме реального времени, необходимо записать примерно следующее:

```
; extconfig.conf
sippeers => odbc,asterisk,ast_sipfriends
```

Чтобы также загружать наших SIP-пользователей из базы данных, задаем следующее:

```
sipusers => odbc,asterisk,ast_sipfriends
```

Вероятно, вы обратили внимание, что и для `sippeers`, и для `sipusers` используется одна и та же таблица. В ней есть поле типа (точно так же, как если бы тип был определен в файле sip.conf), поэтому для извлечения информации мы можем задавать тип `user`, `peer` или `friend`. При описании таблицы для пользователей и равноправных участников необходимо как минимум следующее:

```
+-----+--------+-------+-------+-----+-----------+----------+
|name |host    |secret |ipaddr |port |regseconds | username |
+-----+--------+-------+-------+-----+-----------+----------+
|100  |dynamic |welcome|       |     |1096954152 | 1000     |
+------+-------+-------+-------+-----+-----------+----------+
```

В обязательных полях `port`, `regseconds` и `ipaddr` Asterisk сохраняет регистрационную информацию равноправного участника, чтобы знать, куда направлять вызов. Предполагается, что хост является `dynamic`; однако, если бы равноправный участник был `static`, нам пришлось бы заполнять поле `ipaddr` самостоятельно. Поле `port` является необязательным. Если для него используется стандартный порт, указанный в разделе `[general]`, поле `regseconds` остается пустым. Для SIP-друга можно определить множество других опций, таких как ID вызывающего абонента. Чтобы добавить эту информацию, требуется просто вставить дополнительный столбец `callerid` в таблицу. Другие опции, которые могут быть определены для соединения SIP типа `friend`, можно найти в файле sip.conf.sample.

# Хранение записей параметров вызовов

Записи параметров вызовов (Call Detail Records, CDR) содержат информацию о вызовах, прошедших через систему Asterisk. Более подробно они обсуждаются в главе 13. Хранение CDR – один из самых популярных примеров использования баз данных в Asterisk, потому что в этом случае с CDR проще работать (например, можно отслеживать несколько систем Asterisk в одной таблице).

Создадим в нашей базе данных таблицу для хранения CDR. Зарегистрируемся на сервере PostgreSQL с помощью приложения `psql`:

```
# psql -U asterisk -h localhost asterisk
Password:
```

И создадим таблицу `asterisk_cdr`:

```
asterisk=> CREATE TABLE asterisk_cdr
 (
   id bigserial NOT NULL,
   calldate timestamptz,
```

### Задание параметра systemname для глобальных уникальных идентификаторов

CDR состоят из уникального идентификатора и нескольких полей информации о вызове (включая источник и канал назначения, продолжительность вызова, приложение, выполняемое последним, и т. д.). В кластере серверов Asterisk теоретически возможно дублирование уникальных идентификаторов, поскольку каждая система Asterisk учитывает только саму себя. Чтобы решить эту проблему, можно автоматически добавлять идентификатор системы в начало уникального ID. Для этого введем дополнительную опцию в файл /etc/asterisk/asterisk.conf и зададим идентификатор для каждого из серверов:

```
[options]
systemname=toronto
```

```
    clid varchar(80),
    src varchar(80),
    dst varchar(80),
    dcontext varchar(80),
    channel varchar(80),
    dstchannel varchar(80),
    lastapp varchar(80),
    lastdata varchar(80),
    duration int8,
    billsec int8,
    disposition varchar(45),
    amaflags int8,
    accountcode varchar(20),
    uniqueid varchar(40),
    userfield varchar(255),
    CONSTRAINT asterisk_cdr_id_pk PRIMARY KEY (id)
  )
WITHOUT OIDS;
```

Убедиться в том, что таблица создана, можно с помощью команды \dt (describe tables):

```
asterisk=> \dt asterisk_cdr
            List of relations
Schema  | Name         | Type  | Owner
--------+--------------+-------+----------
public  | asterisk_cdr | table | asterisk
(1 row)
```

Далее сконфигурируем Asterisk на хранение ее CDR в базе данных. Это выполняется в файле /etc/ asterisk/cdr_odbc.conf с помощью следующих настроек:

```
[global]
dsn=asterisk-connector
```

```
username=asterisk
password=welcome
loguniqueid=yes
table=asterisk_cdr
```

Если Asterisk уже запущена, из интерфейса командной строки Asterisk выполняем команду `module reload cdr_odbc.so`. Также можно просто ввести `reload`, чтобы выполнить полную перезагрузку.

```
*CLI> reload
```

Проверим статус CDR. Для этого введем следующую команду и найдем в выводе строку `CDR registered backend: ODBC`:

```
*CLI> cdr status
CDR logging: enabled
CDR mode: simple
CDR registered backend: cdr-custom
CDR registered backend: cdr_manager
CDR registered backend: ODBC
```

Теперь выполним вызов через сервер Asterisk, а потом проверим наличие данных о нем в таблице `asterisk_cdr`. Самый простой способ протестировать вызов – использовать CLI-команду Asterisk `console dial` (предполагается, что имеется звуковая карта и установлен модуль `chan_oss`). Однако для выполнения тестового звонка можно использовать любой имеющийся в распоряжении метод:

```
*CLI> console dial 100@default
-- Executing [100@default:1] Playback("OSS/dsp", "tt-weasels") in new stack
-- <OSS/dsp> Playing 'tt-weasels' (language 'en')
```

Затем установим соединение с базой данных и выполним запрос `SELECT` для проверки наличия данных в таблице `asterisk_cdr`. Также можно выполнить команду `SELECT * FROM asterisk_cdr;`, но в этом случае будет возвращено намного больше данных:

```
# psql -U asterisk -h localhost asterisk
Password:

asterisk=> SELECT id,dst,channel,uniqueid,calldate FROM asterisk_cdr;
id | dst | channel | uniqueid             | calldate
---+-----+---------+----------------------+------------------------
1  | 100 | OSS/dsp | toronto-1171611019.0 | 2007-02-16 02:30:19-05
(1 rows)
```

# Ощутим могущество func_odbc: система «горячих столов»

Функция диалплана `func_odbc` является, наверное, самой замечательной и мощной в Asterisk. Она позволяет создавать и применять довольно простые функции диалплана для извлечения и использования информации из баз данных непосредственно в диалплане. Ее можно использовать в очень многих случаях, например для управления пользо-

вателями или обеспечения возможности совместного использования динамической информации в рамках кластера серверов Asterisk.

`func_odbc` позволяет описывать SQL-запросы и присваивать им имена функций. В результате создаются специальные функции, которые получают свои результаты, выполняя запросы к базе данных. Взаимосвязи между именами создаваемых функций и выражениями SQL, которые они должны выполнять, описываются в файле func_odbc.conf. Используя именованные функции в диалплане, можно извлекать и обновлять значения в базе данных.

Хотя применение внешнего сценария для взаимодействия с базой данных (из которой создается плоский файл для Asterisk) имеет свои преимущества (если база данных дает сбой, система будет продолжать функционировать и сценарий просто не будет обновлять файлы до восстановления соединения с базой данных), основной недостаток в этом случае в том, что любые изменения, вносимые вами для пользователя, остаются недоступными, пока не будет выполнен сценарий обновления. Возможно, это не является большой проблемой для маленьких систем, но в больших системах ожидание вступления в силу внесенных изменений может привести к неприятностям, таким как прерывание активного вызова на время загрузки или синтаксического разбора большого файла.

Ослабить негативные эффекты можно, применяя систему баз данных с дублированием. В версии Asterisk, следующей за 1.4 (в настоящее время эта версия готовится к выпуску[1]) синтаксис файла func_odbc.conf изменится не сильно, но обеспечит возможность в случае отказа перейти к другой СУБД. Таким образом, можно будет кластеризовать серверную часть базы данных, используя отношение ведущий–ведущий (pgcluster; Slony-II) или систему дублирования ведущий–ведомый (Slony-I).

Чтобы вы получили правильное представление о рассматриваемом далее материале, представьте себе дагвудовский сэндвич[2].

Можно ли описать всю гамму впечатлений от такого блюда, только представив изображение помидора или помахав перед носом кусочком сыра? Вряд ли. С этой же проблемой мы сталкиваемся, пытаясь придумать хороший пример, объясняющий, в чем мощь `func_odbc`. Поэтому мы решили «приготовить сэндвич полностью». Рецепт довольно сложен, но, попробовав этот сэндвич, вы не захотите ничего другого.

Для нашего примера мы решили реализовать то, что, по нашему мнению, могло бы иметь практическое применение. Представим небольшую компанию, в отделе продаж которой насчитывается пять человек и им приходится делить между собой два рабочих стола. Это не так ужасно, как

[1] Сейчас уже доступна версия 1.6. – *Примеч. науч. ред.*

[2] А если вы не знаете, что это такое, как раз для этого случая и существует Википедия. Я вовсе не шучу.

может показаться, потому что эти ребята большую часть времени находятся в разъездах и проводят в офисе максимум один день в неделю.

Тем не менее, когда они в офисе, они бы хотели, чтобы система знала, за каким столом они работают, чтобы звонки, адресованные им, направлялись именно туда. Также руководитель хочет иметь возможность отслеживать, когда они находятся в офисе, и управлять привилегиями звонков, производимых с этих телефонов, в их отсутствие.

Решением в такой ситуации, как правило, является использование так называемой системы «горячих столов» (hot-desking). Мы реализовали такую функцию для вас, чтобы продемонстрировать мощь `func_odbc`.

Начнем с простого и создадим два настольных телефона в файле sip.conf.

```
; sip.conf
; ПОЛЬЗОВАТЕЛИ СИСТЕМЫ "ГОРЯЧИХ СТОЛОВ"
[desk_1]
type=friend
host=dynamic
secret=my_special_secret
context=hotdesk
qualify=yes

[desk_2]
type=friend
host=dynamic
secret=my_special_secret
context=hotdesk
qualify=yes

; КОНЕЦ ОПИСАНИЯ ПОЛЬЗОВАТЕЛЕЙ "ГОРЯЧИХ СТОЛОВ"
```

Это два настольных телефона, звонки на которые обрабатываются в контексте `[hotdesk]` файла extensions.conf. Если вы хотите, чтобы эти устройства на самом деле работали, конечно, понадобится задать соответствующие параметры в самих устройствах, но это все рассматривалось в главе 4.

Для файла sip.conf это все. У нас уже есть два кусочка хлеба, но это еще не сэндвич.

Теперь давайте настроим базу данных (предполагаем, что коннектор ODBC базы данных создан и работает, как описывалось в предыдущих разделах данной главы). Сначала подключимся к консоли базы данных следующим образом:

```
# su - postgres
$ psql -U asterisk -h localhost asterisk
Password:
```

Затем, используя следующий фрагмент кода, создадим таблицу:

```
CREATE TABLE ast_hotdesk
(
   id serial NOT NULL,
   extension int8,
```

```
    first_name text,
    last_name text,
    cid_name text,
    cid_number varchar(10),
    pin int4,
    context text,
    status bool DEFAULT false,
    "location" text,
    CONSTRAINT ast_hotdesk_id_pk PRIMARY KEY (id)
)
WITHOUT OIDS;
```

После этого заполняем базу данных следующей информацией (некоторые значения изменятся лишь после выполнения диалплана, но мы приводим их здесь для примера). В консоли PostgreSQL выполните такие команды:

```
asterisk=> INSERT INTO ast_hotdesk ('extension', 'first_name', 'last_name',
'cid_name', 'cid_number', 'pin', 'context', 'location') \
VALUES (1101, 'Leif', 'Madsen', 'Leif Madsen', '4165551101', '555',
'longdistance', 'desk_1');
```

Повторите предыдущую строку и введите собственные значения `VALUES` для всех записей, которые вы желаете видеть в базе данных. Данные таблицы `ast_hotdesk` можно увидеть, выполнив простой запрос `SELECT` из консоли PostgreSQL:

```
asterisk=> SELECT * FROM ast_hostdesk;
```

в результате чего будет получен примерно такой вывод:

```
| id | extension | first_name | last_name      | cid_name           | cid_number   | pin
|--- +-----------+------------+----------------+--------------------+--------------+------
| 1  | 1101      | "Leif"     | "Madsen"       | "Leif Madsen"      | "4165551101" | "555"
| 2  | 1102      | "Jim"      | "Van Meggelen" | "Jim Van Meggelen" | "4165551102" | "556"
| 3  | 1103      | "Jared"    | "Smith"        | "Jared Smith"      | "4165551103" | "557"
| 4  | 1104      | "Mark"     | "Spencer"      | "Mark Spencer"     | "4165551104" | "558"
| 5  | 1105      | "Kevin"    | "Fleming"      | "Kevin Fleming"    | "4165551105" | "559"

| context         | status  | location |$
+-----------------+---------+----------+
| "longdistance"  | "TRUE"  | "desk_1" |
| "longdistance"  | "FALSE" | ""       |
| "local"         | "FALSE" | ""       |
| "international" | "FALSE" | ""       |
| "local"         | "FALSE" | ""       |
```

Теперь у нас есть все ингредиенты, можно приниматься за диалплан. Вот здесь и начинается волшебство.

Прежде чем вы начнете писать программу, обращаем ваше внимание, что весь текст примера можно найти в приложении G. Хотя мы рекомендуем разобрать весь пример вместе с нами по шагам, в приложении вы можете увидеть его целиком (и скопировать его код, если у вас есть электронная версия данной книги).

В файле extensions.conf мы собираемся создать контекст [hotdesk]. Для начала определим шаблонный добавочный номер, который обеспечит возможность пользователям регистрироваться в системе:

```
; extensions.conf
; функция "горячих столов"
[hotdesk]
; Регистрация "горячего стола"
exten => _110[1-5],1,NoOp()
exten => _110[1-5],n,Set(E=${EXTEN})
exten => _110[1-5],n,Verbose(1|Hot Desk Extension ${E} is changing status)
exten => _110[1-5],n,Verbose(1|Checking current status of extension ${E})
exten => _110[1-5],n,Set(${E}_STATUS=${HOTDESK_INFO(status,${E})})
exten => _110[1-5],n,Set(${E}_PIN=${HOTDESK_INFO(pin,${E})})
```

Этот добавочный номер еще не закончен, но прервемся на мгновение и посмотрим, что уже сделано.

Когда агент по продажам занимает рабочий стол, он регистрируется, набирая собственный добавочный номер. В этом случае шаблоном 110[1-5] определены номера от 1101 до 1105. Так же просто можно задать менее жесткое ограничение, используя шаблон 11XX (разрешая номера в диапазоне от 1100 до 1199). Этот добавочный номер использует func_odbc для выполнения поиска с помощью функции диалплана HOTDESK_INFO() (созданием которой мы вскоре займемся). Эта специальная функция (описываемая в файле func_odbc.conf) реализует SQL-выражение и возвращает все, что извлекает из базы данных.

Новая функция HOTDESK_INFO() будет определена в файле func_odbc.conf следующим образом:

```
[INFO]
prefix=HOTDESK
dsn=asterisk
read=SELECT ${ARG1} FROM ast_hotdesk WHERE extension = '${ARG2}'
```

Лишь несколько строк, а так много всего. Давайте быстренько рассмотрим все это, прежде чем двигаться дальше.

Прежде всего, параметр prefix необязательный. Если prefix не задан, Asterisk добавляет в имя функции (в данном случае INFO) префикс ODBC, то есть эта функция будет названа ODBC_INFO(). Такое имя не очень хорошо описывает назначение функции, поэтому полезно задавать префикс, который поможет связать ODBC-функции с задачами, ими выполняемыми. В данном случае мы выбрали имя HOTDESK, то есть данная специальная функция будет названа HOTDESK_INFO.

Атрибут dsn указывает Asterisk, какое из описанных в файле res_odbc.conf соединений использовать. Поскольку в res_odbc.conf может быть сконфигурировано несколько соединений, мы задаем здесь, какое именно должно использоваться. На рис. 12.1 показано отношение между различными настройками файлов и то, как они последовательно ссылаются друг на друга для соединения с базой данных.

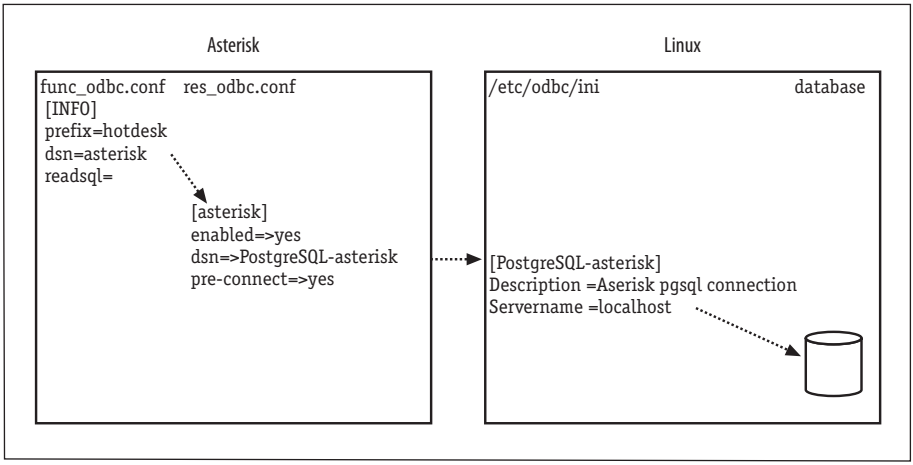



***Рис. 12.1.** Отношения между func_odbc.conf, res_odbc.conf, /etc/odbc.ini (unixODBC) и соединение с базой данных*

Затем описываем SQL-запрос с помощью атрибута `read` (чтение). Существует два разных формата вызова функций диалплана: один – для извлечения информации, а другой – для ее записи. Атрибут `read` используется, когда функция `HOTDESK_INFO()` вызывается в формате для извлечения данных (и можно выполнить отдельное SQL-выражение с атрибутом `write` (запись); формат для атрибута `write` обсуждается в данной главе несколько позже).

Для чтения значений из этой функции в диалплане используется следующий формат:

```
exten => s,n,Set(RETURNED_VALUE=${HOTDESK_INFO(status,1101)})
```

Это обеспечит возвращение значения, расположенного в столбце `status` (статус) базы данных, для которого значение столбца `extension` (добавочный номер) равно `1101`. Переданные в функцию `HOTDESK_INFO()` значения столбца `status` и `1101` помещаются в SQL-запрос, заданный для атрибута `read`, и обозначаются как `${ARG1}` и `${ARG2}`. Если была передана третья опция, она будет доступна как `${ARG3}`.

Убедитесь, что передаваемые данные достаточно уникальны и обеспечивают возвращение только одной строки. Если возвращается несколько строк, Asterisk будет видеть только первую из них. Для PostgreSQL можно ограничить возвращаемые данные одной строкой, добавив в конец SQL-запроса `LIMIT 1`, но это не очень хорошая практика и она не рекомендуется к применению. Чуть дальше в этом разделе мы увидим, как использовать PostgreSQL-функции `LIMIT` и `OFFSET` для перебора нескольких строк данных!

### Использование функции ARRAY()

В нашем примере используется два отдельных вызова базы данных и получаемые в результате значения присваиваются двум переменным канала (${E}_STATUS и ${E}_PIN). Это было сделано с целью упростить пример:

```
exten => _110[1-5],n,Set(${E}_STATUS=${HOTDESK_INFO(status,${E})})
exten => _110[1-5],n,Set(${E}_PIN=${HOTDESK_INFO(pin,${E})})
```

В качестве альтернативы можно было бы возвращать несколько столбцов и сохранять их в разных переменных, используя функцию диалплана ARRAY() (массив). Если SQL-запрос в файле func_odbc.conf определен так:

```
read=SELECT pin,status FROM ast_hotdesk WHERE extension = '${E}'
```

с помощью функции ARRAY() можно в одном обращении к базе данных сохранять каждый столбец данных строки в собственной переменной:

```
exten => _110[1-5],n,Set(ARRAY(${E}_PIN,${E}_STATUS)=${HOTDESK_INFO(${E})})
```

После выполнения SQL-запроса возвращенное значение (если таковое имеется) присваивается переменной канала RETURNED_VALUE (возвращенное значение).

Итак, в первых двух строках следующего фрагмента кода мы передаем значение status и значение, содержащееся в переменной ${E} (например, 1101) в функцию HOTDESK_INFO(). Затем эти два значения замещаются в SQL-запросе на ${ARG1} и ${ARG2} соответственно, SQL-запрос выполняется, а возвращенное значение присваивается переменной канала ${E}_STATUS.

Итак, теперь закончим шаблонный добавочный номер:

```
exten => _110[1-5],n,Set(${E}_STATUS=${HOTDESK_INFO(status,${E})})
exten => _110[1-5],n,Set(${E}_PIN=${HOTDESK_INFO(pin,${E})})
exten => _110[1-5],n,GotoIf($[${ISNULL(${${E}_STATUS})}]?invalid_user,1)
; check if ${E}_STATUS is NULL
exten => _110[1-5],n,GotoIf($[${${E}_STATUS} = 1]?logout,1:login,1)
```

Присвоив значение столбца status переменной ${E}_STATUS (если был набран добавочный номер 1101, имя переменной было бы 1101_STATUS), проверяем, было ли возвращено значение из базы данных (контроль ошибок). Для этой проверки используем функцию ISNULL().

Последняя строка фрагмента кода проверяет статус телефона, и, если в текущий момент он зарегистрирован, будет выполнен его выход из

системы. Если он еще не зарегистрирован, управление перейдет к добавочному номеру login с приоритетом 1 в рамках того же контекста[1].

В следующей после 1.4 версии (в настоящее время готовящейся к выпуску) с выражениями, выполняемыми readsql, можно будет использовать переменную канала ${ODBCROWS}. GotoIf() можно заменить примерно следующим:

```
exten => _110[1-5],n,GotoIf($[${ODBCROWS} < 0]?invalid_user,1)
```

Добавочный номер login выполняет несколько начальных проверок, чтобы убедиться в достоверности введенного агентом кода. Мы предоставляем три попытки для ввода правильного пин-кода. Если все три ввода недействительны, вызов направляется на добавочный номер login_fail (неудачная регистрация) (который будет написан позже).

```
exten => login,1,NoOp() ; задаем исходное значение счетчика
exten => login,n,Set(PIN_TRIES=0) ; задаем максимальное число попыток регистрации
exten => login,n,Set(MAX_PIN_TRIES=3)
exten => login,n(get_pin),NoOp() ; увеличиваем счетчик попыток ввода пин-кода
exten => login,n,Set(PIN_TRIES=$[${PIN_TRIES} + 1])
exten => login,n,Read(PIN_ENTERED|enter-password|${LEN(${${E}_PIN})})
exten => login,n,GotoIf($[${PIN_ENTERED} = ${${E}_PIN}]?valid_login,1)
exten => login,n,Playback(invalid-pin)
exten => login,n,GotoIf($[${PIN_TRIES} <=${MAX_PIN_TRIES}]?get_pin:login_fail,1)
```

Если введен соответствующий пин-код, проверяем регистрационное имя с помощью добавочного номера valid_login (действительное регистрационное имя). Сначала используем переменную CHANNEL (канал), чтобы выяснить, с какого телефона выполняется звонок. Обычно значение переменной имеет примерно такой вид: SIP/desk_1-ab4034c. Поэтому с помощью функции CUT() сначала отбрасываем часть строки SIP/, а оставшееся значение присваиваем переменной LOCATION (местоположение). Затем убираем часть строки -ab4034c, а оставшуюся строку, desk_1, присваиваем переменной LOCATION.

```
exten => valid_login,1,NoOp()
; отбрасываем технологию канала, а оставшуюся строку сохраняем в переменной
; LOCATION
exten => valid_login,n,Set(LOCATION=${CUT(CHANNEL,/,2)})
; отбрасываем уникальный идентификатор и сохраняем оставшуюся строку
```

[1] Помните, что в традиционной телефонной системе все добавочные номера должны быть числовыми, но в Asterisk они могут быть и именованными. Возможное преимущество от применения нечислового добавочного номера в том, что абоненту будет намного сложнее набрать его с обычного телефона, а следовательно, такие номера более безопасны. В этом примере будет использоваться несколько именованных добавочных номеров. Если вы хотите быть абсолютно уверенным, что злонамеренный абонент не сможет дозвониться по этим именованным добавочным номерами, просто используйте прием, применяемый загрузчиком AEL: начинайте обработку не с приоритета 1.

```
; в переменной LOCATION
exten => valid_login,n,Set(LOCATION=${CUT(LOCATION,-,1)})
```

Используем еще одну специальную функцию HOTDESK_CHECK_PHONE_LOGINS(), созданную в файле func_odbc.conf, для проверки, не зарегистрирован ли по этому телефону какой-то другой пользователь. Если количество ранее зарегистрированных пользователей больше 0 (их не может быть больше 1, но мы все равно проводим проверку и сброс для всех вариантов), функция выполняет логику добавочного номера logout_login (отмена регистрации регистрационного имени).

Если никто из агентов не был зарегистрирован ранее, обновляем статус регистрации для этого пользователя с помощью функции HOTDESK_STATUS():

```
exten => valid_login,n,Set(ARRAY(USERS_LOGGED_IN)=${HOTDESK_CHECK_PHONE_
LOGINS(${LOCATION})})
exten => valid_login,n,GotoIf($[${USERS_LOGGED_IN} > 0]?logout_login,1)
exten => valid_login,n(set_login_status),NoOp()

; Задаем для телефона статус '1' - здесь и происходит регистрация
; ПРИМЕЧАНИЕ: здесь надо экранировать запятую, потому что в приложении Set()
; есть аргументы
exten => valid_login,n,Set(HOTDESK_STATUS(${E})=1\,${LOCATION})
exten => valid_login,n,GotoIf($[${ODBCROWS} < 1]?error,1)
exten => valid_login,n,Playback(agent-loginok)
exten => valid_login,n,Hangup()
```

Создаем в файле func_odbc.conf функцию для записи следующим образом:

```
[STATUS]
prefix=HOTDESK
dsn=asterisk
write=UPDATE ast_hotdesk SET status = '${VAL1}', location = '${VAL2}'
WHERE extension = '${ARG1}'
```

Ее синтаксис очень похож на синтаксис read, обсуждаемый ранее в данной главе, но есть несколько новшеств, поэтому рассмотрим их, прежде чем двигаться дальше.

Первое, на что вы, возможно, обратили внимание, – теперь в SQL-запросе есть переменные и ${VAL*x*}, и ${ARG*x*}. Они содержат значения, передаваемые нами в функцию из диалплана. В данном случае имеется две переменных VAL и одна переменная ARG, которые были заданы из диалплана посредством такого выражения:

```
Set(HOTDESK_STATUS(${E})=1\,${LOCATION})
```

Поскольку приложение диалплана Set() может также принимать аргументы (можно задавать множество переменных и значений, разделяя их запятыми или символами вертикальной черты), необходимо экранировать запятую обратным слэшем (\), чтобы синтаксический анализатор выражения обрабатывал ее как часть не приложения Set(), а функции HOTDESK_STATUS().

Обратите внимание, что данный синтаксис немного отличается от синтаксиса функции для чтения. Это говорит Asterisk о том, что необходимо выполнять запись (тот же синтаксис, что и в других функциях диалплана).

Значение переменной ${E} передается в функцию HOTDESK_STATUS(), возвращаемое значение которой затем будет доступно в SQL-выражении в файле func_odbc.conf как переменная ${ARG1}. После этого передаем два значения: 1 и ${LOCATION}. Они доступны SQL-запросу в переменных ${VAL1} и ${VAL2} соответственно.

Как упоминалось ранее, если бы перед тем, как зарегистрироваться в системе, нам пришлось отменить регистрацию одного или более агентов, мы бы реализовывали это с помощью добавочного номера logout_login. В данном фрагменте диалплана для перебора всех строк базы данных и внесения необходимых изменений будет использоваться приложение While(). Скорее всего, цикл будет выполнен только один раз, но это хороший пример того, как можно обновлять или проводить синтаксический разбор множества строк базы данных:

```
exten => logout_login,1,NoOp()
; для всех пользователей, зарегистрированных для данного устройства, задать
; статус "незарегистрирован"
exten => logout_login,n,Set(ROW_COUNTER=0)
exten => logout_login,n,While($[${ROW_COUNTER} < ${USERS_LOGGED_IN}])
```

Переменная ${USERS_LOGGED_IN} была задана ранее функцией HOTDESK_CHECK_PHONE_LOG INS(), которая присвоила ей значение 1 или больше. Мы сделали это, подсчитав количество измененных строк:

```
; func_odbc.conf
[CHECK_PHONE_LOGINS]
prefix=HOTDESK
dsn=asterisk
read=SELECT COUNT(status) FROM ast_hotdesk WHERE status = '1' AND location =
'${ARG1}'
```

Затем с помощью функции HOTDESK_LOGGED_IN_USER() получаем добавочный номер зарегистрированного пользователя. Переменная LOCATION содержит значение desk_1, обозначающее устройство, которое мы хотим проверить, а ${ROW_COUNTER} содержит номер итерации цикла. Оба эти значения передаются как аргументы функции диалплана. Результат затем присваивается переменной WHO (кто):

```
exten => logout_login,n,Set(WHO=${HOTDESK_LOGGED_IN_USER(${LOCATION},
${ROW_COUNTER})})
```

Далее функция HOTDESK_LOGGED_IN_USER() извлекает из базы данных строку, соответствующую итерации цикла, которую мы пытаемся обработать:

```
[LOGGED_IN_USER]
prefix=HOTDESK
dsn=asterisk
read=SELECT extension FROM ast_hotdesk WHERE status = '1'
AND location = '${ARG1}' ORDER BY id LIMIT '1' OFFSET '${ARG2}'
```

Теперь, когда известен добавочный номер, данные для которого требуется изменить, выполняем запись в функцию HOTDESK_STATUS() и присваиваем 0 столбцу status для строки, в которой добавочный номер соответствует значению переменной ${WHO} (то есть 1101). Завершаем цикл с помощью EndWhile() и возвращаемся к добавочному номеру valid_login в приоритет с меткой set_login_status (как обсуждалось ранее):

```
exten => logout_login,n,Set(HOTDESK_STATUS(${WHO})=0)
; отмена регистрации телефона
exten => logout_login,n,Set(ROW_COUNTER=$[${ROW_COUNTER} + 1])
exten => logout_login,n,EndWhile()
exten => logout_login,n,Goto(valid_login,set_login_status)
; возвращаемся к процессу регистрации
```

Все остальное должно быть достаточно понятным (если что-то неясно, вернитесь к главам 5 и 6). Затруднения может вызвать только прием с использованием переменной канала ${ODB CROWS}, которая задается функцией HOTDESK_STATUS(). Она сообщает, сколько строк было изменено в результате SQL-запроса UPDATE (обновить). Мы предполагаем, что это значение равно 1. Если значение ${ODBCROWS} меньше 1, мы рассматриваем это как ошибку и обрабатываем соответствующим образом:

```
exten => logout,1,NoOp()
exten => logout,n,Set(HOTDESK_STATUS(${E})=0)
exten => logout,n,GotoIf($[${ODBCROWS} < 1]?error,1)
exten => logout,n,Playback(silence/1&agent-loggedoff)
exten => logout,n,Hangup()

exten => login_fail,1,NoOp()
exten => login_fail,n,Playback(silence/1&login-fail)
exten => login_fail,n,Hangup()

exten => error,1,NoOp()
exten => error,n,Playback(silence/1&connection-failed)
exten => error,n,Hangup()

exten => invalid_user,1,NoOp()
exten => invalid_user,n,Verbose(1|Hot Desk extension ${E} does not exist)
exten => invalid_user,n,Playback(silence/2&invalid)
exten => invalid_user,n,Hangup()
```

Также включаем контекст hotdesk_outbound, который будет обрабатывать наши исходящие звонки после регистрации агента в системе:

```
include => hotdesk_outbound
```

Контекст hotdesk_outbound преимущественно следует тем же принципам и правилам, которые обсуждались ранее, поэтому не будем рассматривать его слишком подробно. Фактически контекст [hotdesk_outbound] будет обрабатывать все номера, набираемые с настольных телефонов. Сначала задаем переменную LOCATION, используя переменную CHANNEL, затем определяем, какой добавочный номер (агент) зарегистрировался в системе, и сохраняем его в переменную WHO. Если значение этой переменной NULL, отклоняем исходящий звонок. Если значение

переменной не NULL, с помощью функции HOTDESK_INFO() получаем информацию об агенте и сохраняем ее в нескольких переменных канала CHANNEL. Сюда входит и контекст для обработки звонка, где выполняется переход (с помощью функции Goto()) в заданный для нас контекст (который управляет нашим исходящим доступом).

Если попытаться набрать номер, не обрабатываемый нашим контекстом (или одним из промежуточных контекстов – то есть контекст international содержит переход в контекст long distance, который, в свою очередь, содержит переход в local), выполняется встроенный добавочный номер i, который воспроизводит для вызывающего абонента сообщение о невозможности такого действия и отсоединяет его:

```
[hotdesk_outbound]
exten => _X.,1,NoOp()
exten => _X.,n,Set(LOCATION=${CUT(CHANNEL,/,2)})
exten => _X.,n,Set(LOCATION=${CUT(LOCATION,-,1)})
exten => _X.,n,Set(WHO=${HOTDESK_PHONE_STATUS(${LOCATION})})
exten => _X.,n,GotoIf($[${ISNULL(${WHO})}]?no_outgoing,1)
exten => _X.,n,Set(${WHO}_CID_NAME=${HOTDESK_INFO(cid_name,${WHO})})
exten => _X.,n,Set(${WHO}_CID_NUMBER=${HOTDESK_INFO(cid_number,${WHO})})
exten => _X.,n,Set(${WHO}_CONTEXT=${HOTDESK_INFO(context,${WHO})})
exten => _X.,n,Goto(${${WHO}_CONTEXT},${EXTEN},1)

[international]
exten => _011.,1,NoOp()
exten => _011.,n,Set(E=${EXTEN})
exten => _011.,n,Goto(outgoing,call,1)

exten => i,1,NoOp()
exten => i,n,Playback(silence/2&sorry-cant-let-you-do-that2)
exten => i,n,Hangup()

include => longdistance

[longdistance]
exten => _1NXXNXXXXXX,1,NoOp()
exten => _1NXXNXXXXXX,n,Set(E=${EXTEN})
exten => _1NXXNXXXXXX,n,Goto(outgoing,call,1)

exten => _NXXNXXXXXX,1,Goto(1${EXTEN},1)

exten => i,1,NoOp()
exten => i,n,Playback(silence/2&sorry-cant-let-you-do-that2)
exten => i,n,Hangup()

include => local

[local]
exten => _416NXXXXXX,1,NoOp()
exten => _416NXXXXXX,n,Set(E=${EXTEN})
exten => _416NXXXXXX,n,Goto(outgoing,call,1)
```

```
exten => i,1,NoOp()
exten => i,n,Playback(silence/2&sorry-cant-let-you-do-that2)
exten => i,n,Hangup()
```

Если звонок может быть выполнен, он направляется на обработку в контекст `[outgoing]`, где с помощью функции `CALLERID()` задаются имя и номер для ID вызывающего абонента. После этого вызов передается по SIP-каналу с помощью `service_provider`, который был создан в файле sip.conf.

```
[outgoing]
exten => call,1,NoOp()
exten => call,n,Set(CALLERID(name)=${${WHO}_CID_NAME})
exten => call,n,Set(CALLERID(number)=${${WHO}_CID_NUMBER})
exten => call,n,Dial(SIP/service_provider/${E})
exten => call,n,Playback(silence/2&pls-try-call-later)
exten => call,n,Hangup()
```

Наш `service_provider` в файле sip.conf мог бы выглядеть примерно так:

```
[service_provider]
type=friend
host=switch1.service_provider.net
username=my_username
fromuser=my_username
secret=welcome
context=incoming
canreinvite=no
disallow=all
allow=ulaw
```

И это все! Полностью диалплан, используемый для реализации возможности «горячих столов», можно увидеть в приложении G.

Мы только что рассмотрели столько всего, что можно реализовать с помощью `func_odbc`! Теперь вы понимаете, почему нас так восторгает эта функция?!

### Обратная совместимость func_odbc

С Asterisk 1.4 можно использовать версию `func_odbc`, созданную для обеспечения обратной совместимости, которая применяет немного другой формат конфигурации. Это позволяет использовать множество DSN-соединений с разными базами данных, а также применять переменную канала `${ODBCROWS}` для SQL-запросов `read` (`SELECT`). Загрузить версию `func_odbc` для обратной совместимости и установить ее (что приведет к перезаписи существующего файла func_odbc.c) можно так:

```
# cd /usr/src/
# svn co http://svncommunity.digium.com/svn/func_odbc/1.4 ./func_odbc-1.4
# cp func_odbc-1.4/func_odbc.c ./asterisk-1.4/funcs
# cp: overwrite `./asterisk-1.4/funcs/func_odbc.c'? y
```

```
# cd asterisk-1.4
# make install
```

Описанная в данной главе версия работает с текущей версией Asterisk 1.4, но в версии для обратной совместимости (и Asterisk 1.6) будет использоваться следующий измененный синтаксис:

- `read` станет `readsql`.
- `write` станет `writesql`.
- `dsn` станет `readhandle` и `writehandle` (для отдельных строк в базе данных для чтения и записи).
- Может быть перечислено несколько (до 5) `readhandle` и `writehandle`, в порядке предпочтения, для выполнения перехода в случае невозможности соединения с основным обработчиком.
- `prefix` останется неизменным.

Текущий синтаксис, используемый в Asterisk 1.4, будет работать в версии для обратной совместимости, но в следующих версиях будет приводить к формированию предупреждений о том, что этот синтаксис не рекомендуется к использованию. Со временем поддержка такого синтаксиса будет удалена.

## Реализация голосовой почты с использованием ODBC

Asterisk может сохранять голосовую почту в базе данных, используя ODBC-коннектор. Это полезно в кластеризованной среде, когда требуется отделить данные голосовой почты от локальной системы, чтобы обеспечить возможность доступа к одним и тем же данным нескольким серверам Asterisk. Конечно, необходимо учесть, что в этом случае происходит централизация части Asterisk и необходимо предпринять меры для защиты этих данных, такие как регулярное резервное копирование и, возможно, кластеризация серверной части базы данных с помощью дублирования. Для PostgreSQL существует несколько хороших проектов, реализующих это: PGcluster (*http://pgfoundry.org/projects/pgcluster/*) и Slony-I (*http://gborg.postgresql.org/project/slony1/projdisplay.php*).

Asterisk хранит голосовую почту в большом двоичном объекте (Binary Large Object), или BLOB. При извлечении данных она извлекает информацию из BLOB и временно сохраняет ее на жестком диске, пока сообщение воспроизводится для пользователя. Затем, когда пользователь удаляет сообщение голосовой почты, Asterisk удаляет BLOB и соответствующие записи из базы данных. Многие СУБД, такие как MySQL, имеют встроенную поддержку объектов BLOB, но для исполь-

зования этой функциональности в PostgreSQL необходимо предпринять несколько дополнительных шагов, что будет рассмотрено в данном разделе. После этого вы сможете записывать, воспроизводить и удалять данные голосовой почты из базы данных так, как если бы они хранились на локальном жестком диске.

Данный раздел базируется на предыдущих разделах данной главы, посвященных конфигурации. Если вы еще не сделали этого, прежде чем двигаться дальше, обязательно выполните рекомендации разделов «Установка СУБД PostgreSQL» и «Установка и конфигурация ODBC». Выполняя действия, описанные в разделе «Установка и конфигурация ODBC», убедитесь, что активировали опцию ODBS STORAGE (ХРАНИЛИЩЕ ODBS) в разделе Voicemail Build Options (Опции сборки голосовой почты) окна выбора компонентов сборки.

## Создание типа большого объекта

PostgreSQL необходимо показать, как работать с большими объектами. Сюда относится и создание триггера для очистки данных при удалении из базы данных записи, которая ссылается на большой объект.

Установим соединение с базой данных из консоли как пользователь asterisk:

```
# psql -h localhost -U asterisk asterisk
Password:
```

Чтобы создать большой объект, выполним следующий сценарий в консоли PostgreSQL:

```
CREATE FUNCTION loin (cstring) RETURNS lo AS 'oidin' LANGUAGE internal
IMMUTABLE STRICT;
CREATE FUNCTION loout (lo) RETURNS cstring AS 'oidout' LANGUAGE internal
IMMUTABLE STRICT;
CREATE FUNCTION lorecv (internal) RETURNS lo AS 'oidrecv' LANGUAGE internal
IMMUTABLE STRICT;
CREATE FUNCTION losend (lo) RETURNS bytea AS 'oidrecv' LANGUAGE internal
IMMUTABLE STRICT;

CREATE TYPE lo ( INPUT = loin, OUTPUT = loout, RECEIVE = lorecv, SEND = losend,
INTERNALLENGTH = 4, PASSEDBYVALUE );
CREATE CAST (lo AS oid) WITHOUT FUNCTION AS IMPLICIT;
CREATE CAST (oid AS lo) WITHOUT FUNCTION AS IMPLICIT;
```

Для создания функции будем использовать процедурный язык PostgreSQL, называемый pgSQL/PL. Эта функция будет вызываться из триггера, который выполняется при любом изменении или удалении записи из таблицы, применяемой для хранения голосовой почты. Таким образом, происходит очистка данных и в базе данных не остается висячих (несвязанных) строк:

```
CREATE FUNCTION vm_lo_cleanup() RETURNS "trigger"
   AS $$
   declare
     msgcount INTEGER;
   begin
     -- raise notice 'Starting lo_cleanup function for large object with oid
        %',old.recording;[1]
     -- Если это действие обновления, но поле BLOB (lo) не было изменено,
        не делаем ничего
     if (TG_OP = 'UPDATE') then
       if ((old.recording = new.recording) or (old.recording is NULL)) then
          raise notice 'Not cleaning up the large object table,
        as recording has not changed';[2]
         return new;
       end if;
     end if;
     if (old.recording IS NOT NULL) then
       SELECT INTO msgcount COUNT(*) AS COUNT FROM voicemessages WHERE recording
       = old.recording;
       if (msgcount > 0) then
         raise notice 'Not deleting record from the large object table,
         as object is still referenced';[3]
         return new;
       else
         perform lo_unlink(old.recording);
         if found then
           raise notice 'Cleaning up the large object table';[4]
           return new;
         else
           raise exception 'Failed to cleanup the large object table';[5]
           return old;
         end if;
       end if;
   else
     raise notice 'No need to cleanup the large object table, no recording
     on old row';[6]
     return new;
   end if;
end$$
LANGUAGE plpgsql;
```

---

[1] Запускаем функцию `lo_cleanup` для большого объекта с идентификатором объекта %. – *Примеч. перев.*

[2] Не проводим очистку в таблице больших объектов, поскольку записи не менялись. – *Примеч. перев.*

[3] Запись не удаляется из таблицы больших объектов, поскольку по-прежнему существуют ссылки на объект. – *Примеч. перев.*

[4] Очищаем таблицу больших объектов. – *Примеч. перев.*

[5] Не удалось провести очистку в таблице больших объектов. – *Примеч. перев.*

[6] Нет необходимости в очистке таблицы больших объектов, перезапись старой строки не выполняется. – *Примеч. перев.*

Мы собираемся создать таблицу voicemessages (сообщения голосовой почты), в которой будет храниться информация голосовой почты:

```
CREATE TABLE voicemessages
(
   uniqueid serial PRIMARY KEY,
   msgnum int4,
   dir varchar(80),
   context varchar(80),
   macrocontext varchar(80),
   callerid varchar(40),
   origtime varchar(40),
   duration varchar(20),
   mailboxuser varchar(80),
   mailboxcontext varchar(80),
   recording lo,
   label varchar(30),
   "read" bool DEFAULT false
);
```

И теперь надо связать триггер с этой вновь созданной таблицей, чтобы выполнять очистку при любом внесении изменения или удалении из таблицы voicemessages:

```
CREATE TRIGGER vm_cleanup AFTER DELETE OR UPDATE ON voicemessages FOR EACH
ROW EXECUTE
PROCEDURE vm_lo_cleanup();
```

## Конфигурация voicemail.conf для ODBC-хранилища

Чтобы сделать возможным хранение голосовой почты с использованием ODBC, файл voicemail.conf не придется подвергать очень большим изменениям. Фактически в него надо добавить всего три строки! Как правило, в разделе [general] файла voicemail.conf описывается несколько типов форматов, однако нам необходимо определить всего один. Формат wav49 – это формат записи WAV-файлов со сжатием, которые должны воспроизводиться как в настольных системах Linux, так и в Microsoft Windows.

Опция odbcstorage указывает на имя, заданное в файле res_odbc.conf (если вы внимательно читали данную главу, это было имя asterisk). Опция odbctable ссылается на таблицу, в которую должна сохраняться информация голосовой почты. В примерах данной главы использовалась таблица voicemessages:

```
[general]
format=wav49
odbcstorage=asterisk
odbctable=voicemessages
```

Для голосовой почты можно создать отдельный контекст или использовать контекст по умолчанию:

```
[default]
1000 => 1000,J.P. Wiser
```

Теперь подключаемся к консоли Asterisk и выгружаем, а затем повторно загружаем модуль app_voicemail.so:

```
*CLI> module unload app_voicemail.so
== Unregistered application 'VoiceMail'
== Unregistered application 'VoiceMailMain'
== Unregistered application 'MailboxExists'
== Unregistered application 'VMAuthenticate'

*CLI> module load app_voicemail.so
Loaded /usr/lib/asterisk/modules/app_voicemail.so => (Comedian Mail (Voicemail
System))
== Registered application 'VoiceMail'
== Registered application 'VoiceMailMain'
== Registered application 'MailboxExists'
== Registered application 'VMAuthenticate'
== Parsing '/etc/asterisk/voicemail.conf': Found
```

И проверяем успешность загрузки нового почтового ящика:

```
*CLI> voicemail show users for default
Context  Mbox  User         Zone    NewMsg
default  1000  J.P. Wiser               0
```

## Тестирование голосовой почты с использованием ODBC

Создадим простую логику диалплана, чтобы оставлять и извлекать сообщения голосовой почты из тестового ящика голосовой почты. Можно использовать такую простую логику диалплана:

```
[odbc_vm_test]
exten => 100,1,Voicemail(1000@default) ; оставляем сообщение голосовой почты
exten => 200,1,VoicemailMain(1000@default) ; извлекаем сообщение голосовой
                                           ; почты
```

После обновления файла extensions.conf не забудьте перезагрузить диалплан:

```
*CLI> dialplan reload
```

Можно или включить (с помощью выражения `include`) контекст `odbc_vm_test` в контекст, доступный существующему пользователю, или создать отдельного пользователя для тестирования. Если выбран второй вариант, нового SIP-пользователя можно описать в файле sip.conf следующим образом (при условии, что телефон подключен к локальной сети LAN):

```
[odbc_test_user]
type=friend
secret=supersecret
context=odbc_vm_test
host=dynamic
qualify=yes
disallow=all
allow=ulaw
allow=gsm
```

Не забудьте перезагрузить SIP-модуль:

```
*CLI> module reload chan_sip.so
```

и убедиться в том, что SIP-пользователь существует:

```
*CLI> sip show users like odbc_test_user
Username       Secret      Accountcode Def.Context  ACL NAT
odbc_test_user supersecret             odbc_vm_test No RFC3581
```

Затем задаем для телефона или клиента имя пользователя odbc_test_user и пароль supersecret, а потом звоним на добавочный номер 100, чтобы оставить сообщение голосовой почты. В случае успеха вы должны увидеть примерно следующее:

```
-- Executing VoiceMail("SIP/odbc_test_user-10228cac", "1000@default") in new
    stack
  -- Playing 'vm-intro' (language 'en')
  -- Playing 'beep' (language 'en')
  -- Recording the message
-- x=0, open writing: /var/spool/asterisk/voicemail/default/1000/tmp/dlZunm
    format: wav49, 0x101f6534
  -- User ended message by pressing #
  -- Playing 'auth-thankyou' (language 'en')
== Parsing '/var/spool/asterisk/voicemail/default/1000/INBOX/msg0000.txt':
    Found
```

Теперь можно снова использовать приложение `psql` и убедиться, что запись в базу данных действительно осуществляется:

```
# psql -h localhost -U asterisk asterisk
Password:
```

С помощью запроса `SELECT` проверим наличие некоторых данных в таблице `voicemessages`:

```
localhost=# SELECT id,dir,callerid,mailboxcontext,recording FROM voicemessages;
id | dir                          | callerid     | mailboxcontext | recording
---+------------------------------+--------------+----------------+----------
1  | /var/spool/asterisk/         | +18005551212 | default        | 47395
     voicemail/default/1000/INBOX
(1 row)
```

Если запись была помещена в базу данных, должна быть возвращена соответствующая строка. Вы заметите, что в столбце `recording` имеется номер (он, скорее всего, будет отличаться от приведенного здесь), который на самом деле является идентификатором большого объекта, хранящегося в системной таблице. Давайте проверим существование большого объекта в этой системной таблице с помощью команды `lo_list`:

```
localhost=# \lo_list
      Large objects
ID     | Description
-------+-------------
47395  |
(1 row)
```

Проверяем, соответствует ли ID объекта в таблице `voicemessages` указанному в системной таблице больших объектов. Также можно извлечь информацию из базы данных и сохранить ее на жестком диске, чтобы воспроизвести и убедиться, что сообщение было сохранено правильно:

```
localhost=# \lo_export 47395 /tmp/voicemail-47395.wav
lo_export
```

Теперь проверьте аудиозапись, используя свое любимое приложение для работы с аудиоданными, например приложение `play`:

```
# play /tmp/voicemail-47395.wav

Input Filename : /tmp/voicemail-47395.wav
Sample Size    : 8-bits
Sample Encoding: wav
Channels       : 1
Sample Rate    : 8000

Time: 00:06.22 [00:00.00] of 00:00.00 ( 0.0%) Output Buffer: 298.36K

Done.
```

И теперь, когда мы убедились, что все было сохранено в базе данных правильно, можно попытаться прослушать запись с помощью приложения `VoicemailMain()`, позвонив на добавочный номер 200:

```
*CLI>
   -- Executing VoiceMailMain("SIP/odbc_test_user-10228cac", "1000@default")
      in new stack
   -- Playing 'vm-password' (language 'en')
   -- Playing 'vm-youhave' (language 'en')
   -- Playing 'digits/1' (language 'en')
   -- Playing 'vm-INBOX' (language 'en')
   -- Playing 'vm-message' (language 'en')
   -- Playing 'vm-onefor' (language 'en')
   -- Playing 'vm-INBOX' (language 'en')
   -- Playing 'vm-messages' (language 'en')
   -- Playing 'vm-opts' (language 'en')
   -- Playing 'vm-first' (language 'en')
   -- Playing 'vm-message' (language 'en')
== Parsing '/var/spool/asterisk/voicemail/default/1000/INBOX/msg0000.txt':
    Found
```

# Заключение

В данной главе мы рассмотрели несколько областей интеграции Asterisk с реляционной базой данных. Это полезно для систем, в которых необходимо реализовать масштабирование через кластеризацию нескольких серверов Asterisk, работающих с общей централизованной информацией, или когда требуется создавать внешние приложения для изменения информации без перезагрузки системы (то есть без необходимости изменения плоских файлов).

# 13

# Управление системой Asterisk

*Это не будет рассматриваться в книге. Исходный код тоже должен на что-то сгодиться, в конце концов.*

– Ларри Уолл

Вы наверняка полны массой интересных идей, которые собираетесь реализовать с помощью своей новой замечательной системы Asterisk, но есть и некоторые базовые, неинтересные, осмелимся даже сказать, скучные аспекты, которые необходимо обсудить.

## Запись параметров вызовов

Даже без всяких указаний Asterisk предполагает, что вы хотите сохранять информацию CDR[1].

По умолчанию Asterisk создает CSV-файл и помещает его в папку /var/log/asterisk/cdr-csv/[2]. На первый взгляд в этом файле ничего не

---

[1] CDR (Call Detail Record) – запись параметров вызова. В телекоммуникационной сфере это файл, содержащий информацию о работе оборудования, такую как идентификатор источника звонка, идентификатор назначения, длительность и стоимость каждого звонка, общее время работы за тарифицированный период, оставшееся время и списанная за данный период сумма. Формат CDR определяется телекоммуникационным оператором или программой. Некоторые программы позволяют конфигурирование формата CDR пользователем. – *Примеч. науч. ред.*

[2] Файл в формате с разделяющими запятыми (Comma Separated Values, CSV) – общепринятый метод представления данных в текстовом файле. CSV-файл можно открыть в любом текстовом редакторе, но большинство программ обработки крупноформатных таблиц и баз данных также будут читать такие файлы и правильно разбивать их на строки и столбцы.

понятно[1]. Однако, если разделить его на строки соответственно расположению запятых, можно заметить, что в каждой строке содержится информация об отдельном вызове и что запятые разделяют следующие значения:

*accountcode*

Присваивается, если оно задано для канала в конфигурационном файле канала (то есть в sip.conf). Код счета задается для каждого канала. Это значение можно также менять из диалплана, задавая `CDR(accountcode)`.

*src*

Полученный идентификатор вызывающего абонента (строка, 80 символов).

*dst*

Вызываемый добавочный номер.

*dcontext*

Вызываемый контекст.

*clid*

Идентификатор вызывающего абонента с текстом (80 символов).

*channel*

Используемый канал (80 символов).

*dstchannel*

Вызываемый канал, если требуется (80 символов).

*lastapp*

Последнее приложение, если требуется (80 символов).

*lastdata*

Данные последнего приложения (аргументы, 80 символов).

*start*

Начало звонка (дата/время).

*answer*

Ответ на звонок (дата/время).

*end*

Окончание звонка (дата/время).

*duration*

Общая продолжительность в системе, в секундах (целое число), от набора номера до разъединения.

---

[1] Если вас удивляет, почему такая простая на вид вещь считается чем-то особенным, вспомните, что многие традиционные офисные АТС не обладают такой встроенной возможностью. Для этих систем даже для того, чтобы просто фиксировать данные вызовов, не обрабатывая их, необходимо приобретать оборудование сторонних производителей. Asterisk просто сохраняет их. Никаких мучений. Никаких затрат. Никаких шуток.

*billsec*

Общее время вызова, в секундах (целое число), от ответа до разъединения.

*disposition*

Что произошло с вызовом (возможные значения – ANSWERED, NO ANSWER, BUSY).

*amaflags*

Какие флаги использовать (DOCUMENTATION, BILL, IGNORE и т. д.), задаются для каждого канала в отдельности, как *accountcode*. AMA расшифровывается как Automated Message Accounting (автоматизированные учетно-расчетные операции услуг связи). AMA-флаги являются стандартными (предположительно) в отрасли.

*userfield*

Определенное пользователем поле, максимум 255 символов.

**Хранение CDR в базе данных**

CDR также могут храниться в базе данных. В настоящее время Asterisk поддерживает СУБД SQLite, PostgreSQL, MySQL и unixODBC, но в данной книге рассматривается только ODBC (см. главу 12). Многие предпочитают хранить CDR в базе данных, чтобы управлять счетами и ресурсами с помощью запросов.

# Работа с журналами регистрации

В ходе работы Asterisk формирует события, которые будут обусловливать создание записи или в основных системных журналах регистрации, или в собственных файлах журнала Asterisk. В сильно загруженной системе (или системе, в которой возникла существенная проблема) эти файлы могут быстро достигать очень больших размеров. Если включить отладку, процессы, участвующие в записи файлов журнала, могут начать оказывать влияние на производительность системы. По умолчанию Asterisk будет просто наращивать эти файлы до тех пор, пока жесткий диск не будет полностью заполнен. К счастью, Linux предоставляет утилиты для реализации чередования файлов журнала (чтобы ни один из файлов не разрастался слишком сильно), а также удаления старых файлов журнала (что не даст засорить ими систему).

Утилита logrotate обычно запускается операционной системой раз в день[1]. К сожалению, без специального сценария, который проинс-

[1] Подробнее о работе с утилитой logrotate и настройке ее параметров можно узнать с помощью команды man logrotate. – *Примеч. науч. ред.*

труктирует logrotate, как работать с Asterisk, файлы журнала Asterisk будут разрастаться без всякого контроля. Чтобы этого не произошло, необходимо настроить параметры для Asterisk в специальном файле в папке /etc/logrotate.d. Этот файл должен будет сменить текущий файл журнала и послать Asterisk инструкции о необходимости смены журнала Asterisk (что заставит ее прекратить использовать теперь уже устаревший журнал и сформировать новый файл).

Создадим новый файл /etc/logrotate.d/asterisk и поместим в него следующие строки:

```
/var/log/asterisk/* /var/log/asterisk/cdr-csv {
missingok
sharedscripts
monthly
rotate 12
postrotate
   asterisk -rx "logger rotate" > /dev/null 2> /dev/null
endscript
}
```

Этот файл указывает утилите logrotate чередовать журналы Asterisk каждый месяц, сохраняя их в течение 12 месяцев. По завершении этого периода утилита сообщает Asterisk, что все файлы журнала были использованы (это заставит Asterisk создать новые файлы журнала и начать записывать информацию в них). Мы выбрали эти значения произвольно, вы можете назначить собственные соответственно своим нуждам.

## Выполнение Asterisk под учетной записью пользователя, не обладающего правами администратора

По умолчанию Asterisk выполняется под учетной записью администратора (root), и, хотя мы не располагаем никакой официальной информацией, из собственного опыта мы пришли к выводу, что подавляющее большинство систем Asterisk работают в этом стандартном состоянии, используемом по умолчанию. С точки зрения безопасности это недопустимо рискованно. Странно, но кажется, что большинство из нас готовы рисковать[1].

Запустить Asterisk под учетной записью пользователя, не обладающего правами администратора, не так уж сложно, но требует нескольких

[1] Для многих пользователей так удобнее, поскольку с правами root в Linux можно выполнять практически любые операции. Однако с точки зрения безопасности это недопустимо. Поэтому крайне желательно выполнить те рекомендации по установке, которые описаны далее в этой главе.

дополнительных шагов. Кроме того, отладка может привести в уныние, если не понимать принцип распределения прав доступа в Linux. Но с точки зрения безопасности это стоит затраченных усилий.

Мы собираемся запустить Asterisk под учетной записью пользователя asterisk, поэтому необходимо сначала создать этого пользователя в нашей системе. Следующие команды будут выполняться под учетной записью администратора. Мы скажем, когда перейти к использованию учетной записи asterisk, которую собираемся сейчас создать:

```
# adduser -c "Asterisk PBX" asterisk
# passwd asterisk
```

Теперь, создав пользователя asterisk, переключимся на этого пользователя, под учетной записью которого будем выполнять все остальные команды. Как только мы переключились на пользователя asterisk с помощью команды `su`[1], можно загружать копию Asterisk через SVN, FTP или WGET, компилировать и устанавливать ее. В следующем примере мы собираемся взять копию Asterisk из хранилища SVN.

1.4.5 – текущая выпущенная версия на момент написания данной книги. Пока книга готовилась к публикации, могла выйти новая версия, поэтому проверьте последнюю версию на веб-сайте Asterisk. Другими словами, не надо бездумно вводить 1.4.5 в своем коде, копируя наши примеры. Выясните, какова текущая версия, и используйте ее.

```
# su - asterisk
$ svn co http://svn.digium.com/svn/asterisk/tags/1.4.5 asterisk-1.4.5
$ cd asterisk-1.4.5
$ ./configure --prefix=$HOME/asterisk-bin --sysconfdir=$HOME/asterisk-bin
   --localstatedir=$HOME/asterisk-bin
$ make menuselect
$ make install
```

Выполняя сценарий `./configure` с флагом `--prefix`, мы указываем системе установить двоичные компоненты в подпапку asterisk-bin папки `$HOME`[2]. Флаг `--sysconfdir` указывает системе, куда поместить конфигурационные файлы, а флаг `--localstatedir` – куда установить дополнительные файлы, например звуковые. Главное здесь то, что, поскольку загрузка, компиляция и установка выполняются от лица пользова-

---

[1] Буквосочетание `su` исторически означает super-user (суперпользователь), но теперь оно также могло бы означать switch-user (другой пользователь) или substitute-user (пользователь-заместитель). Символ `-` в команде указывает `su` использовать среду для этого пользователя (например, использовать `PATH` для этого пользователя).

[2] `HOME` – системная переменная, определяющая путь к домашней папке для текущего пользователя, то есть по умолчанию /home/asterisk.

теля asterisk, все, что создается, будет закреплено за этим пользователем и будет иметь права, предоставленные этому пользователю.

Теперь можно установить файлы примеров, тоже в папку $HOME/asterisk-bin/asterisk:

```
$ make samples
```

Выполним тестовый запуск Asterisk с помощью следующей команды:

```
$ ./asterisk-bin/sbin/asterisk -cvvv
```

Обычно Asterisk должна выполняться как сервис. Во время установки команда `make config` устанавливает сценарии инициализации. К сожалению, этого не произойдет, если вы зарегистрированы как пользователь asterisk, потому что только пользователь root имеет право изменять команды запуска системы. Кажется, что надо зарегистрироваться как root, перейти к папке /home/asterisk/asterisk-1.4.5 и снова выполнить команду `make config` (теперь с правами, которые позволят действительно ее выполнить). И проблема решена, не так ли?

Да, но не совсем. Если вы выполните команду `service asterisk start`, система сообщит, что не может найти asterisk. Знаете, почему? Потому, что сценарий инициализации считает, что исполняемый файл asterisk должен быть установлен в /usr/sbin, где он должен был бы находиться, если бы мы установили asterisk под учетной записью root. Итак, мы должны сообщить сценарию инициализации, где искать asterisk и сценарий `safe_asterisk`:

```
# ln -s /home/asterisk/asterisk-bin/sbin/asterisk /usr/sbin/asterisk
# ln -s /home/asterisk/asterisk-bin/sbin/safe_asterisk /usr/sbin/safe_asterisk
```

Поскольку наш сценарий инициализации использует сценарий `safe_asterisk` и по умолчанию пытается запустить Asterisk под учетной записью администратора, необходимо изменить сценарий `safe_asterisk`, указав ему запускать Asterisk под учетной записью пользователя, не обладающего правами администратора. Итак, откроем сценарий `safe_asterisk` в своем любимом текстовом редакторе и найдем переменную `ASTARGS` (это примерно 78-я строка). Вставим в кавычки `-U asterisk`:

```
#
# Don't fork when running "safely"[1]
#
ASTARGS="-U asterisk"
```

Пойдем дальше и запустим Asterisk, выполнив команду `service asterisk start`. С помощью команды `ps` убедимся, что Asterisk запущена под учетной записью пользователя asterisk:

```
# service asterisk start
# ps aux | grep asterisk
```

[1] Не выполнять ветвления при запуске в безопасном режиме «safely» – *Примеч. перев.*

```
503 30659 0.0 1.8 26036 8692 pts/2 Sl 15:07 0:00
/home/asterisk/asterisk-bin/sbin/asterisk -U asterisk -vvvg -c
```

Фактически 503 – это наш пользователь asterisk; проверяем это, заглянув в файл /etc/passwd:

```
# cat /etc/passwd

asterisk:x:503:503:Asterisk PBX:/home/asterisk:/bin/bash
```

Выполним перезагрузку системы, чтобы убедиться, что все идет, как надо. Помните, что многие операции, осуществляемые с Asterisk, могут предполагать ее выполнение под учетной записью администратора, поэтому необходимо следить за возможными ошибками, касающимися ограниченных прав доступа. Ваш процесс Asterisk может считать себя суперпользователем, но мы немножко подрезали ему крылышки.

Зачем создавать проблемы? Суть в том, что если какое-либо слабое место системы безопасности Asterisk[1] обеспечит кому-либо доступ к серверу через учетную запись asterisk, его возможности в системе будут ограничены правами данной учетной записи. Если Asterisk выполняется под учетной записью root, нарушение правил безопасности обеспечивает злоумышленнику полный контроль над вашей системой.

## Настройка голосовых сообщений системы

Благодаря своей практически безграничной гибкости Asterisk можно менять голосовые сообщения системы. Это очень просто объяснить, но, как правило, сложно хорошо выполнить.

Основной дистрибутив включает 300 голосовых сообщений системы, плюс еще 600 в дополнении asterisk-sounds. Если вы задумали настроить их все, необходимо иметь или кучу денег, или массу времени.

Звукотехнику рекомендуется обеспечить нормализацию всех записей до –3 дБ и их корректное начало и завершение (с соответствующей паузой в начале и конце сообщения).

Имея в распоряжении готовые записи, применить их легко – просто замените файлы в папке /var/lib/asterisk/sounds теми, которые вы создали.

---

[1] Если зайти в любую систему, выполняющую Asterisk с настройками безопасности по умолчанию, подключить к ней клавиатуру и монитор и нажать сочетание клавиш Alt+F9, вы подключитесь к интерфейсу командной строки Asterisk. Нажмите клавиши ! и Return – и вы зашли в оболочку. Если Asterisk запущена от имени пользователя root, вы теперь владеете этой системой.

### Голос

Если вас интересует, кто является «голосом» Asterisk, то ее зовут Эллисон Смит (Allison Smith). Она может записать специальные сообщения и для вашей системы.

Это мощная концепция, потому что очень немногие офисные АТС предоставляют возможность использовать один и тот же голос и в собственных, и в системных сообщениях.

Эллисон – «голос» сообщений системы как на английском, так и на испанском языках.

Также доступны сообщения на французском. «Голосом» французского Asterisk является Джун Уоллак (June Wallack) из Montreal's own (она также озвучивает сообщения на безупречном английском[1], если вы захотите, чтобы сообщения на обоих языках были записаны одним голосом).

Чтобы узнать, как получить собственные голосовые сообщения, посетите веб-сайт Digium по адресу *http://www.digium.com/products/voice*.

В качестве альтернативы можно записать сообщения самостоятельно и поместить их любую папку по своему выбору. В приложениях `Playback()` или `Background()` можно указывать полный путь к звуковым файлам или любую подпапку /var/lib/asterisk/sounds/.

Обратите внимание, что стандартные звуки Asterisk записаны в формате GSM. Хранить их в этом формате целесообразно, только если система будет взаимодействовать со множеством каналов, использующих кодек GSM). Конечно, файлы в формате GSM занимают меньше места на жестком диске, но, на наш взгляд, дополнительная нагрузка на ЦП для их преобразования (не говоря уже о более низком качестве звука в целом) делает нежелательным применение GSM. При использовании файлов без сжатия (таких, как .wav, .ulaw или .alaw) ЦП не придется так напряженно работать. И как дополнительный приз – лучшее качество звучания сообщений.

---

[1] Мы хотели сказать «на английском без акцента», но тогда нам пришлось бы извиняться перед жителями Британских островов, Австралии, Южной Африки и прочими. Мы не являемся специалистами в языках, диалектах и т. п., но можем различить акцент, свойственный профессиональным дикторам Северной Америки. Этот акцент распространен на Тихоокеанском побережье от Сан-Диего до Сиэтла, а также в большей части англоговорящей Канады. Джун и Эллисон говорят с таким акцентом, и, по нашему мнению, это звучит прекрасно.

### Запись звука из диалплана

Удивительно, но один из самых простых способов получить записи приличного качества – не с помощью ПК с невообразимым ПО для редактирования, а через телефонный аппарат. На это есть масса причин, но самое важное то, что телефон, как правило, обеспечивает запись без фонового шума (такого, как белый шум, создаваемый оборудованием отопления, вентиляции и кондиционирования воздуха) и с постоянным уровнем громкости.

Это небольшое дополнение в диалплан позволит без труда создавать записи, которые будут помещаться в папку /tmp/ вашей системы (где их можно переименовать, а затем перенести в любое другое место):

```
exten => _66XX,1,Wait(2)
exten => _66XX,n,Record(/tmp/prompt${EXTEN:2}:wav)
exten => _66XX,n,Wait(1)
exten => _66XX,n,Playback(/tmp/prompt${EXTEN:2})
exten => _66XX,n,Wait(2)
exten => _66XX,n,Hangup()
```

Данный фрагмент кода обеспечит возможность звонить на номера от 6600 до 6699 и записывать сообщения в папку /tmp/ под именами от prompt00.wav до prompt99.wav. После завершения записи (о чем вы сообщите, нажав кнопку #) ваше сообщение будет воспроизведено и соединение завершится.

Не забудьте перенести свои сообщения из папки /tmp/ в папку для звуковых файлов Asterisk. Чтобы диалплан оставался понятным, присвойте файлам prompt<XX> более осмысленные имена. Например:

```
mv /tmp/prompt00.wav /var/lib/asterisk/sounds/custom/welcome-message.wav
```

## Музыка во время ожидания

Любая широко распространенная система АТС предлагает возможность воспроизведения музыки во время ожидания ответа. В этом вопросе Asterisk предоставляет широкое поле для творчества.

Сегодня все хорошо знакомы с музыкальным форматом MP3 и использование MP3-файлов в качестве источника музыки во время ожидания представляет большой интерес. Конечно, это кажется хорошей идеей, но есть некоторые вопросы, над которыми, по нашему мнению, следует задуматься:

- MP3-файлы чрезвычайно сложны, и их декодирование налагает серьезную нагрузку на ЦП. Если имеется много каналов, извлекающих музыку из системы (например, иногда люди любят слушать

музыку по телефону или сразу несколько абонентов ожидают ответа информационно-справочной службы), требования к ЦП сильно возрастают, что связанно с необходимостью воспроизведения хранящихся MP3-файлов. Компьютер, который в других условиях обеспечивал бы нужды системы, может оказаться непригодным.

- Современные жесткие диски могут хранить большие объемы данных, поэтому, вероятно, нет никакой причины беспокоиться о сокращении пространства, занимаемого звуковыми файлами. Сжатие звука имеет смысл с точки зрения передачи по сетям (MP3-файл намного меньше и удобнее для скачивания, чем эквивалентный ему файл в формате .wav), но в рамках системы на самом деле неважно, сколько места занимают эти файлы.
- При использовании MP3-файлов обычно возникают проблемы с лицензированием. ;-)

Учитывая все это, рекомендуем преобразовать используемые музыкальные файлы в собственный формат различных кодеков, которые поддерживает ваша система. Например, если для внутренних телефонов поддерживается μlaw и G.729 в VoIP-сетях, музыку лучше хранить в обоих форматах, чтобы Asterisk не приходилось выполнять перекодировки для воспроизведения музыки в этих каналах.

Мы часто используем в своих системах общедоступную музыку (или музыку, лицензированную Creative Commons). Музыка Creative Commons

### Свободно распространяемая музыка

Многие не знают, что воспроизведение музыки во время ожидания требует специальной лицензии, даже если вы используете музыку с приобретенного вами CD или передаваемую по радио. Во избежание недоразумений мы рекомендуем вам использовать только музыку, предлагаемую в музыкальной индустрии, которая не обременена никакими видами лицензий.

Существует множество веб-сайтов, на которых можно найти музыку, лицензированную так, что она подходит для использования в качестве музыки во время ожидания. Мы нашли два таких сайта: *http://en.wikipedia.org/wiki/Wikipedia:Sound/list* и *http://www.opsound.org/*.

Оба предлагают достаточно большую музыкальную коллекцию, которую можно без труда скачать. Заметьте, что не все музыкальные файлы профессионального качества, поэтому обязательно прослушайте их, прежде чем включать в свою коллекцию музыки во время ожидания[1].

[1] Чтобы найти больше свободно распространяемой музыки, выполните поиск по словам Creative Commons music (музыка Creative Commons).

часто поступает в формате ogg-Vorbis (который концептуально аналогичен MP3, но несовместим с ним). Чтобы воспроизводить файлы .ogg или .mp3 в системе Asterisk, необходимо преобразовать их в формат, с которым может работать Asterisk. Для этого сделаем следующее:

1. Убедимся, что установлена утилита SoX (Sound eXchange). Если нет, выполняем следующую команду для ее установки:

   ```
   $ yum install sox
   ```

2. Скачаем выбранную музыку в рабочую папку в своей системе (/tmp, пожалуй, подходящее место). Например, следующая команда обеспечивает загрузку красивой фортепьянной музыки композитора Пахельбеля:

   ```
   $ wget http://upload.wikimedia.org/wikipedia/commons/6/62/Pachelbel%27s_Canon.ogg
   ```

3. Теперь надо преобразовать эту запись из формата ogg-Vorbis в более подходящий для Asterisk формат:

   ```
   $ sox Pachelbel\'s_Canon.ogg -r 8000 -c 1 -s -w moh1.wav resample -ql
   ```

   Возможно, понадобится также скорректировать амплитуду с помощью опции -v.

   Итак, мы взяли исходный файл, преобразовали его в WAV-файл, подходящий для Asterisk[1] и сохранили получившийся в результате файл как moh1.wav.

4. Практически все сделано. Осталось только создать папку для постоянного хранения новых файлов (/tmp, конечно, не место для них):

   ```
   $ mkdir /var/lib/asterisk/mohwav
   ```

   и перенести их туда:

   ```
   $ mv *.wav /var/lib/asterisk/mohwav
   ```

5. Поскольку мы разместили наши музыкальные файлы в другой папке, а не в той, куда Asterisk устанавливает свои образцы музыки, необходимо внести изменения в конфигурационный файл, чтобы отразить это. Редактируем файл /etc/asterisk/musiconhold.conf следующим образом:

   ```
   [default]
   mode=files
   directory=/var/lib/asterisk/mohwav
   random=yes
   ```

Скажем несколько слов о том, какую музыку использовать. Это будет зависеть от того, какое впечатление вы желаете произвести на своих абонентов. Независимо от вашего выбора, следует помнить, что:

---

[1] Обратите внимание, что можно использовать любой формат, совместимый с Asterisk; .wav был выбран в данном примере лишь потому, что ЦП легко выполнять преобразования μlaw/alaw/slin «на лету» и при этом с ним также легко работать в других средах.

- Люди на самом деле не хотят ждать, поэтому обычно они не планируют слишком долго оставаться в таком состоянии. Это означает, что нет особого смысла давать им прослушивать серьезные музыкальные произведения. Их мысли сейчас заняты другим, и у них не будет времени проникнуться этой музыкой.
- Телефонная система не обеспечивает качества, необходимого для точного воспроизведения звуков. Низкие басы обычно звучат ужасно, а высокие частоты воспринимаются просто как шум. Музыка должна быть простой, тогда ей, скорее всего, обеспечено хорошее звучание.
- Музыкальные предпочтения различных людей очень разнообразны, и, хотя может показаться хорошей идеей охватить широкий диапазон стилей, такая эклектичная музыкальная подборка вызовет скорее раздражение, чем расположение.

Классическая музыка удовлетворяет всем перечисленным выше критериям, и ее легко достать. Также она обладает превосходным звучанием (и не без основания!), так что это наиболее приемлемый выбор, хотя, следует признать, под эту музыку не хочется пуститься в пляс.

В пакете для скачивания Asterisk вместе с исходным кодом имеется три композиции, лицензированных для использования с Asterisk. Эти композиции являются лишь образцами. Поскольку их только три, они быстро надоедят людям, которые будут часто вам звонить. Только в кошмарном сне можно себе представить, что Asterisk обязана своим

### Случайный выбор музыки во время ожидания

В традиционной офисной АТС музыка во время ожидания поступает из одного источника. В один момент времени все слышат одну и ту же музыку, и даже если никого нет в режиме ожидания, музыка все равно воспроизводится. В Asterisk музыка не воспроизводится, если в ней нет необходимости, и каждому вызывающему абоненту предоставляется собственный источник музыки. Если бы Asterisk просто начинала воспроизводить композиции в том порядке, в каком их находит, для каждого вызова, переведенного в режим ожидания, всегда проигрывалась бы одна и та же мелодия с начала. Чтобы смоделировать традиционное поведение при задержке вызова, Asterisk может (и обычно должна) быть настроена на воспроизведение музыки в случайном порядке. Это означает, что система будет случайным образом выбирать файл для воспроизведения. Если в каталоге музыки во время ожидания достаточно различных композиций, можно свести к минимуму вероятность того, что тот, кто звонит вам часто, будет вынужден постоянно слушать одну и ту же мелодию.

всемирным успехом радости людей от прослушивания одних и тех же трех мелодий при ожидании ответа на звонок. Поэтому мы написали этот раздел.

## Заключение

Данная глава могла бы стать книгой (и, возможно, однажды так и произойдет). Мы выбрали для рассмотрения несколько тем, которые, на наш взгляд, представляют ценность для большинства читателей, но, несомненно, тем для обсуждения намного больше. Как и многие вопросы в данной книге, они были затронуты лишь в самых общих чертах.

# 14

# Попурри

*На 90% задачи уходит 90% времени,*
*а на оставшиеся 10% уходит еще 90% времени.*

– Правило девяносто к десяти

Самым сложным при создании данной книги было выбрать не то, о чем писать, а то, о чем не будет возможности рассказать. Теперь, когда базовые вопросы рассмотрены, вы готовы к тому, чтобы услышать правду: мы вовсе не научили вас всему тому, что необходимо знать об Asterisk.

Теперь, пожалуйста, поймите, что это не потому, что мы не хотели дать все самое лучшее, а лишь из-за того, что Asterisk безгранична (во всяком случае, нам так кажется).

В данной главе мы хотим дать вам почувствовать вкус некоторых диковинок, которые Asterisk приберегла для вас. Практически каждый раздел этой главы мог бы стать отдельной книгой (и они *станут* книгами, если Asterisk добьется того успеха, который мы ей предвещаем).

## Festival

Festival – популярный механизм речевого воспроизведения текста с открытым исходным кодом. Основная идея использования Festival с Asterisk заключается в том, что диалплан может передавать тело текста в Festival, который будет «читать» этот текст вызывающему або-

ненту. Вероятно, самое очевидное применение Festival – чтение электронной почты вслух, когда вы в дороге[1].

## Настройка и подготовка Festival к работе с Asterisk

В настоящее время существует два способа использования Festival с Asterisk. Первый (самый простой) метод, без доработки и повторной компиляции Festival, – загрузить в конфигурационный файл Festival следующий текст (festival.scm обычно располагается в папке /etc/ или /usr/share/festival/):

```
(define (tts_textasterisk string mode)
"(tts_textasterisk STRING MODE)

Apply tts to STRING. This function is specifically designed for use in server
mode so a single function call may synthesize the string. This function name
may be added to the server safe functions."[2]
(let ((wholeutt (utt.synth (eval (list 'Utterance 'Text string)))))
(utt.wave.resample wholeutt 8000)
(utt.wave.rescale wholeutt 5)
(utt.send.wave.client wholeutt)))
```

Этот текст можно поместить в любом месте файла, но только так, чтобы он не попал в другие круглые скобки.

Второй (и более традиционный) способ – скомпилировать Festival со специальным патчем для Asterisk (располагающимся в подпапке contrib/ папки исходного кода Asterisk).

Информация по обоим методам имеется в файле README.festival, который находится в подпапке contrib/ папки исходного кода Asterisk.

Для любого метода вам придется изменить список доступа Festival в файле festival.scm. Просто выполните поиск по слову localhost и замените его полным доменным именем своего сервера.

Оба метода настраивают Festival так, чтобы он мог правильно взаимодействовать с Asterisk. Настроив Festival, необходимо запустить сервер Festival. После этого можно вызывать приложение `Festival()` из диалплана.

---

[1] Пожалуй, самый замечательный пример применения Festival – в ZoIP Симона Дитнера (Simon Ditner). Это порт популярной игры Zork с механизмом полной поддержки речи, работающим в Asterisk (ZoIP также использует Sphinx, но это не будет рассматриваться в данной книге). Мы собираемся придумать новое название для таких вещей. Это не видеоигра, поскольку нет экрана; вероятно, поэтому она должна называться аудиоигрой. Найти ее и оценить всю ее прелесть можно по адресу *http://www.zoip.org*.

[2] Применяем tts к STRING. Эта функция специально разработана для использования в режиме сервера, чтобы можно было синтезировать строку одним вызовом функции. Это имя функции может быть добавлено в список безопасных функций сервера. – *Примеч. перев.*

## Конфигурация Asterisk для работы с Festival

Конфигурационный файл Asterisk, обеспечивающий настройки для работы с Festival, называется festival.conf, что вполне логично. В этом файле задаются имя хоста и порт используемого сервера Festival, а также некоторые настройки для кэширования речи, генерируемой Festival. Для большинства установок (если вы собираетесь запускать Festival на своем сервере Asterisk) прекрасно подходят настройки по умолчанию.

## Запуск сервера Festival

Чтобы запустить сервер Festival для отладки, просто выполните команду `festival` с аргументом `--server`:

```
[root@asterisk ~]# festival --server
```

Убедившись в том, что сервер Festival работает и не отклоняет ваших соединений, можно запустить Festival, введя следующее:

```
[root@asterisk ~]# festival_server 2>&1 >/dev/null &
```

## Вызов Festival из диалплана

Теперь, когда Festival сконфигурирован и сервер Festival запущен, организуем его вызов в простом диалплане:

```
exten => 123,1,Answer()
exten => 123,2,Festival(Asterisk and Festival are working together)
```

Перед вызовом `Festival()` всегда должно быть вызвано приложение `Answer()`, чтобы гарантировать установление соединения по каналу.

Когда Asterisk соединяется с Festival, на терминале, с которого был запущен сервер Festival, должен появиться такой вывод:

```
[root@asterisk ~]# festival --server
server    Sun May 1 18:38:51 2005 : Festival server started on port 1314
client(1) Sun May 1 18:39:20 2005 : accepted from asterisk.localdomain
client(1) Sun May 1 18:39:21 2005 : disconnected
```

### Еще один способ использования Festival с Asterisk

Некоторые участники сообщества разработчиков Asterisk сообщают о том, что им удалось передать текст в утилиту Festival text2wave и воспроизвести в Asterisk результирующий WAV-файл. Например, это можно сделать так:

```
exten => 124,1,Answer()
exten => 124,2,System(echo "Это проверка Festival" | /usr/bin/text2wave
-scale 1.5 -F 8000 -o /tmp/festival.wav)
```

```
exten => 124,3,Playback(/tmp/festival)
exten => 124,4,System(rm /tmp/festival.wav)
exten => 124,5,Hangup()
```

Этот метод также позволяет вызывать другие механизмы речевого воспроизведения текста, такие как популярный речевой механизм производства компании Cepstral (*http://www.cepstral.com*), который является недорогой коммерческой производной Festival с очень приятными голосами. Для этого примера будем считать, что Cepstral установлен в папку /usr/local/cepstral/:

```
exten => 125,1,Answer()
exten => 125,2,System(/usr/local/cepstral/bin/swift -o /tmp/swift.wav
¨Это проверка Cepstral¨)
exten => 125,3,Playback(/tmp/swift)
exten => 125,4,System(rm /tmp/swift.wav)
exten => 125,5,Hangup()
```

Появление следующего вывода означает, что в список доступа в файле festival.scm не был добавлен хост, вследствие чего соединение было отклонено:

```
[root@asterisk ~]# festival --server
server    Sun May 1 18:30:52 2005 : Festival server started on port 1314
client(1) Sun May 1 18:32:32 2005 : rejected from asterisk.localdomain not
in access list
```

# Файлы вызовов

Файлы вызовов позволяют создавать вызовы в оболочке Linux. Эти мощные события запускаются путем размещения файла .call в папке /var/spool/asterisk/outgoing/. Фактически имя файла не имеет значения, но хорошей практикой является давать ему информативное имя и заканчивать его расширением .call.

Когда файл вызова появляется в папке исходящих вызовов, Asterisk практически немедленно начинает действовать согласно содержащимся в нем инструкциям[1].

Файлы вызовов записываются в следующем формате. Сначала определяем, куда будем звонить:

```
Channel: канал
```

Можно задать время ожидания ответа на звонок (по умолчанию 45 с), время между повторными попытками дозвониться и максимальное число попыток. Если параметр `MaxRetries` (максимальное число попыток) опущен, выполняется только одна попытка вызова:

[1] Речь здесь идет о миллисекундах. Не верите? Проверьте сами!

```
WaitTime: число
RetryTime: число
MaxRetries: число
```

Если ответ на звонок получен, здесь мы определяем, где он должен обрабатываться:

```
Context: имя-контекста
Extension: добавочный номер
Priority: приоритет
```

В качестве альтернативы можно задать только приложение и передавать аргументы в него:

```
Application: Playback()
Data: hello-world
```

Далее задаем Caller ID (ID звонящего) исходящего звонка:

```
CallerID: Asterisk 800-555-1212
```

Задаем переменные канала следующим образом:

```
SetVar: john=Zap/1/5551212
SetVar: sally=SIP/1000
```

и добавляем код учетной записи CDR:

```
Account: документация
```

Нельзя создавать файл вызова из папки, в которой находится очередь. Asterisk активно отслеживает подкачку и попытается захватить файл даже еще до того, как он будет закончен! Файлы вызовов должны создаваться в какой-то другой папке, потом в той же папке создается копия этого файла и эта копия с помощью команды `mv` перемещается в папку подкачки. Заметьте, что мы назвали команду `mv`, не `cp`. Это важно, потому что процесс копирования в Linux реализован таким образом, что файл появляется в папке назначения еще до того, как он оказывается там полностью. В противоположность этому, использование операции `mv` не позволит файлу появиться в папке назначения до полного завершения операции перемещения. При копировании очень велика вероятность того, что Asterisk начнет читать файл до того, как он весь будет перенесен туда, что приведет к непредвиденным результатам.

# DUNDi

Если бы возникли опасения, что Марк Спенсер может исчерпать свой запас интересных идей, система Distributed Universal Number Discovery (DUNDi) с легкостью пресекла бы их. DUNDi является такой же революционной разработкой, как и Asterisk. Лучшее определение дано на веб-сайте DUNDi (*http://www.dundi.com*): «DUNDi™ – это одноранговая система для поиска интернет-шлюзов в сервисы телефонии. В отличие

от традиционных централизованных сервисов (таких, как необыкновенно простой и лаконичный стандарт ENUM; *http://www.faqs.org/rfc/rfc2916.txt*), DUNDi является полностью распределенным и вообще не имеет никакой централизованной службы». DUNDi – это в некотором роде протокол маршрутизации для VoIP.

## Как работает DUNDi

DUNDi можно рассматривать как большую телефонную книгу, которая позволяет запрашивать у равноправных участников сети альтернативный VoIP-маршрут к добавочному номеру или телефонному номеру PSTN.

Например, предположим, вы подключены к сети DUNDi-test (это бесплатная и открытая сеть, которая обеспечивает звонки на традиционные номера PSTN). Вы спрашиваете своего друга Боба, знает ли он, как связаться с номером 1-212-555-1212, к которому у вас нет прямого доступа. Боб отвечает: «Я не знаю, как позвонить на этот номер, но сейчас спрошу у своей подружки Салли (которая является равноправным участником сети)».

Боб спрашивает Салли, не знает ли она, как связаться с требуемым номером, и она отвечает: «С этим номером можно связаться по адресу IAX/dundi:очень_длинный_пароль@имяхоста/добавочный номер». Боб сохраняет этот адрес в своей базе данных и передает вам информацию о том, как связаться с 1-800-555-1212 через VoIP, предоставляя альтернативный метод достижения той же цели по другой сети.

Поскольку Боб сохранил найденную информацию, он сможет предоставлять ее всем равноправным участникам сети, которые будут запрашивать этот же номер у него позже, таким образом, им не придется продолжать свои поиски. Это способствует снижению нагрузки на сеть и сокращает время ответа для часто запрашиваемых номеров. (Однако следует отметить, что DUNDi создает циклически сменяющийся ключ, таким образом, хранящаяся информация остается действительной лишь ограниченный период времени.)

DUNDi выполняет поиск динамически или с помощью выражения `switch =>` в вашем файле extensions.conf, или используя приложение `DUNDiLookup()`. DUNDi доступен только в Asterisk версии 1.2 и выше.

DUNDi-протокол может использоваться и в локальной сети. Скажем, вы администратор системы Asterisk в очень большой корпорации и желаете упростить процесс управления добавочными номерами. В этой ситуации мог бы использоваться DUNDi, обеспечивая возможность нескольким серверам Asterisk (предположительно расположенным в разных офисах компании и объединенным в одноранговую сеть) выполнять динамические поиски VoIP-адресов добавочных номеров в сети.

## Конфигурация Asterisk для использования с DUNDi

Для работы с DUNDi необходимо сконфигурировать три файла: dundi.conf, extensions.conf и iax.conf[1]. Файл dundi.conf управляет аутентификацией равноправных участников, которым мы разрешаем выполнять поиск в нашей системе. Этот файл также содержит список равноправных участников сети, которым мы можем направлять свои запросы поиска. Поскольку на одном сервере могут выполняться несколько разных сетей, для каждого равноправного участника необходимо определить собственный раздел и затем сконфигурировать сети, в которых ему разрешено выполнять поиски. Кроме того, необходимо определить, каких равноправных участников мы желаем использовать для осуществления поисков.

### Общее пиринговое соглашение

Общее пиринговое соглашение, или General Peering Agreement (GPA), – это имеющее обязательную юридическую силу лицензионное соглашение, разработанное для предотвращения злоупотреблений с протоколом DUNDi. Перед подключением к группе DUNDi-test необходимо подписать GPA. GPA используется для защиты членов группы и для установления между ними доверительных отношений. Обязательным требованием группы DUNDi-test является указание полной и точной контактной информации в файле dundi.conf, чтобы остальные участники одноранговой группы могли связаться с вами. GPA можно найти в подпапке doc/ папки исходного кода Asterisk.

### Общая конфигурация

Раздел `[general]` файла dundi.conf содержит параметры, относящиеся к общим вопросам работы клиента и сервера DUNDi:

```
; конфигурационный файл DUNDi
;
[general]
;
department=IT
organization= toronto.example.com
locality=Toronto
stateprov=ON
country=CA
email=support@toronto.example.com
phone=+19055551212
;
```

[1] Обязательно должны быть сконфигурированы файлы dundi.conf и extensions.conf. Мы выбрали iax.conf для предоставления информации о нашем адресе по сети, но DUNDi является протоколо-независимым, и таким образом, вместо iax.conf мы могли бы использовать sip.conf, h323.conf или mgcp.conf.

```
; Задаем адрес привязки и номер порта. ПО умолчанию - 4520
;bindaddr=0.0.0.0
port=4520
entityid=FF:FF:FF:FF:FF:FF
ttl=32
autokill=yes
;secretpath=dundi
```

Идентификатор объекта, заданный `entityid`, вообще должен быть адресом управления доступом к устройству (Media Access Control, MAC) интерфейса компьютера. По умолчанию в качестве идентификатора объекта используется первый Ethernet-адрес сервера, но его можно переопределить с помощью `entityid`, если ему присвоен MAC-адрес *какого-то* принадлежащего вам устройства. Рекомендуется использовать MAC-адрес основного внешнего интерфейса. С помощью этого адреса другие равноправные участники будут идентифицировать вас.

Поле Time To Live (`ttl`) определяет длину цепочки равноправных участников, от которых мы желаем получать ответы, и используется для прерывания циклов опроса. При каждой передаче запроса вниз по цепочке участников до тех пор, пока запрашиваемый номер не будет найден, значение поля TTL увеличивается на единицу, точно так же как это происходит с полем TTL пакета ICMP (Internet Control Message Protocol – межсетевой протокол контрольных сообщений, используемый для отладки и мониторинга в IP-сетях). Поле TTL также определяет максимальную продолжительность (в секундах) ожидания ответа.

Если вам необходимо найти номер, вашим равноправным участникам сети рассылается исходный запрос (называемый `DPDISCOVER`) об этом номере. Если вы не получаете подтверждение приема (`ACK`) своего запроса (`DPDISCOVER`) в течение 2000 мс (время, достаточное только для передачи сигнала) и параметр `autokill` имеет значение `yes`, Asterisk разошлет равноправным участникам сообщение `CANCEL` (отменить). (Заметьте, что подтверждение приема необязательно является ответом на запрос; это лишь подтверждение того, что участник получил запрос.) Назначение `autokill` – предотвращение задержек поиска из-за хостов с большим временем ожидания. Кроме опций `yes` и `no`, можно также задавать время ожидания в миллисекундах.

Модуль `pbx_dundi` создает циклически сменяющийся ключ и сохраняет его в локальной базе данных Asterisk (AstDB). Имя ключа `secret` хранится в семействе `dundi`. Значение ключа можно увидеть с помощью команды `database show` в консоли Asterisk. Семейство базы данных может быть переопределено опцией `secretpath` (путь к базе ключей).

### Создание отображающихся контекстов

Файл dundi.conf определяет контексты DUNDi, отображаемые в контексты диалплана в файле extensions.conf. Контексты DUNDi – это способ описания особых и отдельных групп служб каталогов. Контексты

раздела отображения указывают на контексты файла extensions.conf, управляющего номерами, информацию о которых вы предоставляете. При создании равноправного участника вы должны определить, по каким отображающимся контекстам он может выполнять поиск. Делается это с помощью выражения `permit` (у каждого равноправного участника может быть несколько выражений `permit`). Отображающиеся контексты связаны с контекстами диалплана в том смысле, что они являются границей зоны безопасности для ваших равноправных участников.

Информация о телефонных номерах должна предоставляться в следующем формате:

```
<код_страны><код_города><префикс><номер>
```

Например, полный североамериканский номер был бы представлен так: 14165551212.

Все отображающиеся контексты DUNDi принимают форму

```
dundi_контекст => локальный_контекст,вес,технология,местоназначения[,опции]]
```

Приведенная ниже конфигурация создает отображающийся контекст DUNDi, который мы будем использовать для предоставления группе DUNDi-test информации о наших локальных телефонных номерах. Обратите внимание, что все это должно располагаться в одной строке:

```
dundi-test => dundi-local,0,IAX2,dundi:${SECRET}@toronto.example.com/
${NUMBER},nounsolicited,nocomunsolicit,nopartial
```

В этом примере отображающийся контекст – это `dundi-test`. Он указывает на контекст `dundi-local` в файле extensions.conf (предоставляющий список телефонных номеров, звонки с которых он обрабатывает). Для номеров офисной АТС *вес* равен нулю (соединение выполняется напрямую). Вес номера, отличный от 0, свидетельствует о наличии нескольких переходов или путей на маршруте достижения места назначения. Это значение может быть полезно при получении нескольких ответов на один поиск; предпочтительным будет путь с меньшим весом.

Если мы можем ответить на запрос поиска, наш ответ будет содержать метод, с помощью которого другой конец линии сможет соединиться с системой. Сюда относится используемая технология (такая, как IAX2, SIP, H.323 и т. д.), имя пользователя и пароль, по которым выполняется аутентификация, хост, на который следует отправлять аутентификационную информацию, и наконец добавочный номер.

Asterisk предоставляет несколько сокращенных записей, что позволяет создавать «шаблон», по которому можно построить наши ответы. В шаблоне могут использоваться следующие переменные канала:

`${SECRET}`

Замещается шаблоном, хранящимся в локальной AstDB.

`${NUMBER}`

Запрашиваемый номер.

`${IPADDR}`

IP-адрес для соединения.

Обычно безопаснее статически конфигурировать имя хоста, а не использовать переменную `${IPADDR}`. Переменная `${IPADDR}` иногда предоставляет адрес в частном пространстве IP-адресов, недоступном из Интернета.

### Описание равноправных участников DUNDi

Равноправные участники DUNDi описываются в файле dundi.conf и идентифицируются уникальным MAC-адресом второго уровня интерфейса удаленной системы. Файл dundi.conf – это то место, где мы определяем, в каком контексте выполнять поиск для равноправных участников, запрашивающих поиск, и каких равноправных участников мы хотим использовать при выполнении поиска для конкретной сети:

```
[00:00:00:00:00:00] ; Удаленный офис
model = symmetric
host = montreal.example.com
inkey = montreal
outkey = toronto
include = dundi-test
permit = dundi-test
qualify = yes
dynamic=yes
```

Идентификатор удаленного равноправного участника (MAC-адрес) заключается в квадратные скобки ([ ]). `inkey` и `outkey` – это пара ключей (открытый и закрытый), используемых для аутентификации. Пары ключей генерирует сценарий `astgenkey`, располагающийся в подпапке ./asterisk/contrib/scripts/ папки исходного кода. Не забывайте использовать флаг `-n`, чтобы не приходилось создавать пароли при каждом запуске Asterisk:

```
# cd /var/lib/asterisk/keys
# /usr/src/asterisk/contrib/scripts/astgenkey -n toronto
```

Полученные в результате ключи, toronto.pub и toronto.key, будут помещены в папку /var/lib/asterisk/keys/. Файл toronto.pub – это открытый ключ, который должен быть отправлен веб-серверу, чтобы он был доступен всем участникам, с которым вы желаете установить одноранговую связь. При установлении одноранговой связи вы можете передать равноправным участникам открытый ключ, доступный по протоколу HTTP, который они могут поместить в свои папки /var/lib/asterisk/keys/.

После загрузки ключей необходимо повторно загрузить модули res_crypto.so и pbx_dundi.so в Asterisk:

```
*CLI> module reload res_crypto.so
   -- Reloading module 'res_crypto.so' (Cryptographic Digital Signatures)
   -- Loaded PRIVATE key 'toronto'
   -- Loaded PUBLIC key 'toronto'

*CLI> module reload pbx_dundi.so
   -- Reloading module 'pbx_dundi.so' (Distributed Universal Number Discovery
(DUNDi))

== Parsing '/etc/asterisk/dundi.conf': Found
```

Затем в файле iax.conf создаем пользователя dundi, чтобы обеспечить возможность соединения с вашей системой Asterisk. После аутентификации вызова запрашиваемый добавочный номер передается в контекст `dundi-local` файла extensions.conf, где выполняется его обработка.

### Обеспечение возможности удаленных соединений

Вот описание канала типа `user` для пользователя dundi:

```
[dundi]
type=user
dbsecret=dundi/secret
context=dundi-local
disallow=all
allow=ulaw
allow=g726
```

Вместо использования статического пароля Asterisk повторно создает пароль каждые 3600 с (1 ч). Это значение сохраняется в /dundi/secret базы данных Asterisk и предоставляется посредством переменной `${SECRET}`, описанной в отображающемся контексте в файле dundi.conf. Увидеть текущие ключи для всех равноправных участников, включая свои локальные открытый и закрытый ключи, можно, выполнив команду `show keys` в интерфейсе командной строки Asterisk.

Запись `context=dundi-local` определяет контекст в extensions.conf, в который направляются прошедшие авторизацию вызывающие абоненты. Оттуда мы можем обрабатывать звонок, точно так же как делали бы это в диалплане любого другого входящего соединений.

### Конфигурация диалплана

Файл extensions.conf определяет, информацию о каких номерах вы предоставляете и что делаете с вызовами, адресованными им. Контекст `dundi-local` выполняет две задачи:

- Управляет номерами, информацию о которых мы предоставляем. Они указаны в отображающемся контексте `dundi` в файле dundi.conf.
- Определяет то, что должно быть сделано с вызовом, указанным в описании пользователя dundi в iax.conf.

Для предоставления информации о диапазонах номеров и управления входящими вызовами используется мощная возможность сопоставления с шаблонами, доступная в диалплане. В следующем диалплане предоставляется информация только о номере +1-416-555-1212, но так же просто можно применить сопоставление с шаблоном для предоставления информации о диапазоне номеров или добавочных номеров:

```
[dundi-local]
exten => 14165551212,1,NoOp(dundi-local: Number advertisement and incoming)
exten => 14165551212,n,Answer()
exten => 14165551212,n(call),Dial(SIP/1000)
exten => 14165551212,n,Voicemail(u1000)
exten => 14165551212,n,Hangup()
exten => 14165551212,n(call)+101,Voicemail(b1000)
exten => 14165551212,n,Hangup()
```

# Альтернативные методы хранения голосовой почты

Обычный способ хранения голосовой почты Asterisk – простая запись сообщения в файл, размещаемый на локальном жестком диске в папке /var/spool/asterisk/voicemail. Этот метод хорош для простых офисных АТС, но в больших распределенных сетях или окружениях, в которых желательна более тесная интеграция с внешними приложениями, потребуются более сложные методы хранения.

## Хранение голосовой почты на IMAP-сервере

Уже довольно долгое время телефонная отрасль обещает предоставить возможность хранения голосовых сообщений вместе с обычными сообщениями электронной почты. Это называется универсальной системой передачи и обработки сообщений (Unified Messaging), и хотя большинство офисных АТС сегодня предлагают такую систему в том или ином виде, обычно ее лицензирование и реализация очень дороги.

Естественно, Asterisk отбрасывает все эти глупости и просто позволяет интегрировать ящик голосовой почты в среду IMAP. Хранение голосовой почты на IMAP-сервере обеспечивает несколько преимуществ. После прослушивания по телефону сообщение голосовой почты переводится в состояние read (прочитано) на IMAP-сервере. Это означает, что клиентское почтовое приложение также отметит это сообщение как прочитанное. Аналогичным образом, если сообщение прослушивается из клиентского почтового приложения, система голосовой почты отключает сигнал уведомления о наличии непрочитанного сообщения на всех телефонах, закрепленных за этим почтовым ящиком. Удаление сообщения в одном месте обеспечит его удаление везде. Таким образом, однажды удаленное сообщение действительно уничтожается. Такова универсальная система передачи и обработки сообщений, «свя-

щенный грааль» интеграции голосовой и электронной почты, но Asterisk скромно предпочитает не акцентировать на этом особое внимание.

Интеграция с IMAP по-прежнему является новой функциональностью, поэтому для ее активации необходимо кое-что добавить. Первым делом для работы с IMAP-сервером Asterisk необходимо установить IMAP-клиент. Подойдет практически любой IMAP-сервер (даже Exchange Server). Авторы лично протестировали поддержку голосовой почты по протоколу IMAP с IMAP-серверами Courier-IMAP и Dovecot. IMAP-сервер физически может располагаться на одном компьютере с установленной Asterisk или находиться в другой части земного шара. Чтобы иметь возможность доступа к IMAP-серверу, Asterisk требует наличия библиотеки IMAP-клиента. Эта библиотека является бесплатным IMAP-клиентом под именем c-client, созданным в Вашингтонском университете. Чтобы установить c-client, надо просто перейти в свою папку /usr/src и выполнить следующие команды:

```
# wget ftp://ftp.cac.washington.edu/mail/imap.tar.Z
```

Это обеспечит загрузку исходного кода. Извлеките его из архива следующим образом:

```
# tar zxvf imap.tar.Z
```

Обратите особое внимание на имя папки, создаваемой этой командой, поскольку оно, вероятно, изменится к тому времени, когда вы будете читать данную книгу. Пока книга готовилась к выходу, имя папки менялось четыре раза. Последним было /usr/src/imap-2006h.

Перейдите в созданную папку и выполните команду

```
# make lrh IP6=4
```

Это обеспечит компиляцию всего, что необходимо Asterisk для использования библиотек IMAP-клиента[1].

Теперь необходимо повторно скомпилировать Asterisk с возможностями IMAP. Понадобится перейти в ту папку, в которой располагаются исходные файлы Asterisk (к примеру, /usr/src/asterisk), и выполнить следующую команду:

```
# /configure --with-imap=/usr/src/imap-2006h
```

Теперь необходимо повторно выполнить команду `make menuconfig`, чтобы включить IMAP-хранилище в компиляцию. В разделе Voicemail Build Options (Параметры создания голосовой почты) выберите параметр IMAP STORAGE (ХРАНИЛИЩЕ IMAP) и нажмите кнопку x, чтобы сохранить изменения и выйти. Теперь при компиляции Asterisk будет выполнена и сборка

[1] Опция `lrh` указывает компилятору, что это система Linux Red Hat. Опция `IP6=4` говорит о том, что мы не хотим выполнять компиляцию с поддержкой IPv6. Остальные опции можно найти в Makefile. Для систем RHEL 5 или CentOS 5 вместо `lrh` необходимо использовать `lr5`.

модуля IMAP. Очевидно, что следующий шаг – повторная компиляция и установка Asterisk. Для этого просто выполните в вашем терминале следующую команду:

```
# make && make install
```

Итак, модуль скомпилирован и установлен. Пришло время его сконфигурировать.

Добавим несколько строк в раздел `[general]` файла voicemail.conf, хранящегося в папке /etc/asterisk:

```
imapserver=localhost
imapport=143
expungeonhangup=yes
authuser=vmail
authpassword=vmailsecret
imapfolder=Voicemail
```

Поскольку Dovecot доступен в хранилище пакетов CentOS, установить небольшой IMAP-сервер для обработки виртуальных пользователей (голосовой почты) на сервере Asterisk просто:

```
# yum install dovecot
```

Обеспечим активацию поддержки IMAP в файле /etc/dovecot.conf, раскомментировав строку `protocols`, чтобы она выглядела следующим образом:

```
protocols = imap imaps
```

Активировав поддержку IMAP, создадим учетную запись пользователя для сохранения почты:

```
# groupadd vmail
# useradd vmail -g vmail -s /bin/true -c "asterisk voicemail user" -p
  vmailsecret -d /var/spool/asterisk/imap-voicemail vmail
# chown -R vmail.vmail /var/spool/asterisk/imap-voicemail
```

Теперь перезапустим Dovecot и Asterisk – и все должно быть готово.

```
# service dovecot restart
# service asterisk restart
```

Поздравляем! Вы успешно установили базовую поддержку голосовой почты IMAP в Asterisk! Однако это лишь верхушка айсберга. Имея хранилище голосовой почты, работающее по протоколу IMAP, можно без труда реализовать совместно используемую (например, по отделам) голосовую почту с помощью совместно используемых IMAP-папок. Многие компании уже имеют совместно используемую по отделам электронную почту, так что совместно используемый ящик голосовой почты – весьма естественное и логичное развитие технологии. При наличии IMAP-хранилища голосовой почты каждый сотрудник может работать с несколькими ящиками голосовой почты, не смешивая личные и рабочие сообщения. С точки зрения Asterisk в конфигурации нет ничего необычного; вы просто вызываете приложение `VoiceMail()` с указанием желаемого почтового ящика и контекста и обеспечиваете, что-

бы совместно используемая IMAP-папка была включена в список папок почтового клиента служащего отдела.

Наконец, вместо глобального IMAP-пользователя Asterisk может быть реализована авторизация для каждого почтового ящика в отдельности (то есть каждый ящик голосовой почты аутентифицируется как отдельный пользователь). Asterisk поддерживает это с помощью опций `imapuser` и `imapsecret` в индивидуальных описаниях ящиков голосовой почты:

```
[imapvoicemail]
100 => 1234,Sue's Mailbox,,,imapuser=sue@example.tld|imapsecret=suesimapsecret
101 => 5555,Bob's Mailbox,,,imapuser=bob@example.tld|imapsecret=bobsimapsecret
```

В этом конкретном примере, если сообщение оставлено в почтовом ящике IMAP 100 контекста `imapvoicemail`, Asterisk будет проходить аутентификацию на IMAP-сервере как sue@example.tld, используя в качестве пароля suesimapsecret. Аналогично, если сообщение оставлено в почтовом ящике 101 того же контекста голосовой почты, будут использоваться соответственно bob@example.tld и bobimapsecret.

## Хранение голосовой почты в базе данных ODBC

На всякий случай повторяю, что голосовая почта также может храниться в базе данных посредством ODBC-коннектора. Этому посвящена глава 12!

# Asterisk и Jabber (XMPP)

Jabber – на самом деле первоначальное имя протоколов IETF XMPP (RFC 3920-3923). Поскольку имя Jabber, несомненно, лучше, чем XMPP, оно и закрепилось. Этот протокол изначально разрабатывался для обеспечения децентрализованной, общедоступной инфраструктуры обмена сообщениями и контроля присутствия, поддерживающей открытые стандарты. Она поддерживает доставку сообщений в автономном режиме и шифрование и «доросла» до голосовых сообщений, которые поддерживает Asterisk.

Интересно отметить, что вначале Jabber рассматривался как конкурент протокола SIMPLE, основывающегося на SIP. XMPP разработан как более универсальный протокол и, конечно, на основе XML.

Asterisk можно сконфигурировать на использование XMPP в нескольких видах. XMPP может выступать в роли инфраструктуры контроля присутствия (например, добавочный номер 205 не отвечает или занят) или инфраструктуры обмена голосовыми сообщениями JINGLE для полной поддержки телефонной связи с другими сервисами, такими как Google Talk.

В отличие от других сетей обмена сообщениями, таких как MSN и Yahoo!, XMPP децентрализована. Кто угодно может иметь свой собс-

твенный Jabber-сервер и выполнять на нем любое количество сервисов. Сообщения отправляются практически так же, как сообщения электронной почты: используемый Jabber-сервер связывается с Jabber-сервером другого человека и устанавливается прямое соединение. Если этот человек не в сети, сообщение сохраняется, и, когда он регистрируется на своем Jabber-сервере, выполняется доставка всех хранящихся сообщений. Учитывая возможность шифрования (протокол XMPP поддерживает TLS), неудивительно, что многие коммерческие организации переходят к реализации внутренних сетей обмена сообщениями с использованием этого замечательного протокола. И Asterisk может органично интегрироваться в эти сети связи.

## Заключение

Вот и все, что мы собирались рассказать в данной главе, но это далеко не все, что вы можете узнать об Asterisk. Надеемся, вы начинаете осознавать, насколько велики возможности Asterisk.

В следующей главе мы заглянем в будущее телекоммуникаций и обсудим, почему (и как глубоко) мы верим, что Asterisk прекрасно подходит на главную роль в нем.

# 15

# Asterisk – будущее телефонии

*Сначала они не замечают вас, потом осмеивают, затем начинают бороться и наконец – проигрывают.*
– Махатма Ганди

Вот мы и добрались до последней главы книги. Рассмотрено многое, но, надеемся, теперь вы понимаете, что мы лишь слегка коснулись этого феномена, называемого Asterisk. Чтобы прийти к логическому завершению, давайте рассмотрим, что могут предложить Asterisk и системы телефонной связи с открытым исходным кодом в ближайшем будущем.

Прогнозы – неблагодарное дело, но мы абсолютно уверены, что появление механизмов связи с открытым исходным кодом, таких как Asterisk, предвещает изменения в мышлении, которые приведут к трансформации всей телекоммуникационной отрасли. В данной главе мы представим основания нашей веры в эти идеи.

## Проблемы традиционной системы телефонной связи

Чаще всего создателем телефона называют Александра Грэхема Белла, но на самом деле во второй половине 19 века десятки умов работали над задачей передачи голоса по телеграфным линиям. Эти люди в большинстве своем были ребятами с деловой жилкой, жаждущими создать продукт, который помог бы им сколотить состояние.

Мы привыкли думать о телефонных компаниях как о монополиях, но это было совсем не так на заре их существования. История телефонной связи начиналась в среде с очень высокой конкуренцией, новые компании боролись за рынок по всему свету, зачастую не обращая внимания на нарушение авторских прав. Многие известные монополии начинали с участия (и победы) в патентных войнах.

Интересно сравнить историю телефонии с историей Linux и Интернета. Телефон был создан как коммерческий проект, и развитие телефонной отрасли шло через судебные тяжбы и поглощение компаний, однако Linux и Интернет возникли из академического сообщества, в котором обмен знаниями всегда был превыше коммерческой выгоды.

Культурные различия очевидны. Телекоммуникационные технологии стремятся к закрытости, запутанности и дороговизне, тогда как сетевые технологии, как правило, открыты, хорошо задокументированы и конкурентоспособны.

## Закрытость мышления

Если сравнивать культуру телекоммуникационной отрасли с Интернетом, иногда сложно поверить, что они имеют родство. Интернет был создан энтузиастами, тогда как индивидуальное участие в разработке PSTN невозможно себе представить. Это «клуб для избранных»; членство в нем доступно не каждому[1].

Международный союз телекоммуникаций (International Telecommunication Union, ITU), несомненно, демонстрирует такой тип закрытого мышления. Тот, кто хочет прикоснуться к его знаниям, должен быть готов заплатить за них. Членство требует подтверждения квалификации, и вам придется заплатить тысячи долларов за доступ к его библиотеке публикаций.

Хотя ITU – организация, санкционированная ООН и отвечающая за международную связь, многие протоколы VoIP (SIP, MGCP, RTP, STUN) вышли не из священных стен ITU, а были ратифицированы IETF (который публикует все свои стандарты абсолютно бесплатно и принимает на рассмотрение любые проекты спецификаций интернет-протоколов).

Открытые протоколы, такие как SIP, могут иметь тактическое преимущество над ITU-протоколами, например H.323, благодаря доступности. Хотя H.323 популярен среди поставщиков услуг связи как базовый VoIP-протокол, попробуйте найти терминалы на базе H.323. Новые продукты чаще всего поддерживают SIP.

---

[1] Сравните с клубной страницей IETF, которая гласит: «IETF – не организация со специальными атрибутами членства (никаких карточек, никаких взносов, никаких секретных рукопожатий :-)... Она открыта для всех, кому это интересно... Добро пожаловать в IETF». Вот это сообщество!

Успех открытого подхода IETF не остался незамеченным могущественным ITU. Не так давно на веб-сайте ITU появилась возможность бесплатно скачать до трех документов[1]. Очевидно, что идея открытости появилась в их умах. Недавние заявления ITU свидетельствуют о желании достичь «большей вовлеченности в ITU гражданского общества и академического мира». Г-н Хоулинь Чжао (Houlin Zhao), директор Бюро по стандартизации телекоммуникаций (Telecommunication Standardization Bureau, TSB) ITU, верит, что «ITU должен предпринять шаги, которые способствовали бы этому»[2].

План достижения этой открытости неясен, но ITU начинает осознавать неизбежное.

Что касается Asterisk, она охватывает и прошлое, и будущее: доступна поддержка H.323, хотя сообщество преимущественно отказалось от H.323 в пользу IETF-протокола SIP и любимчика сообщества Asterisk IAX.

## Несоблюдение стандартов

Самая большая странность всех существующих в мире традиционных телекоммуникаций стандартов – кажущаяся неспособность различных производителей реализовать их единообразно. Каждый производитель желает достичь глобальной монополии, таким образом, концепция совместимости отходит на задний план в погоне занять первое место на рынке с новой идеей.

ISDN-протоколы – классический пример этому. Развертывание ISDN было (и во многом остается и сейчас) болезненным и дорогим предприятием, поскольку каждый производитель решил реализовать его немного по-своему. ISDN мог бы обеспечить появление масштабной общедоступной сети передачи данных на 10 лет раньше создания Интернета. К сожалению, из-за цены, сложности и проблем совместимости ISDN не пошел дальше передачи голоса и лишь в редких случаях используется для транспортировки видео или данных теми людьми, кто готов платить. ISDN довольно широко распространен (особенно в Европе и в Северной Америке в реализациях крупных АТС), но он и близко не реализует те возможности, которые мог бы.

По мере все большего распространения VoIP необходимость в ISDN исчезнет.

## Длительные циклы выпуска новых версий

Серьезным людям нужны месяцы, а иногда и годы, чтобы принять новое направление, не говоря уже о выпуске продукта, совместимого с ним.

---

[1] Учитывая, что таких документов тысячи и каждый из них обычно ссылается на десятки других, трудно говорить о ценности такой свободной информации.

[2] *http://www.itu.int/ITU-T/tsb-director/itut-wsis/files/wg-wsis-Zhao-rev1.pdf*

Кажется, что, прежде чем новая технология может быть утверждена и только запланирована для разработки, она должна быть проанализирована вдоль и поперек и затем еще успешно пройти все бюрократические процедуры. Выхода любого полезного продукта можно ожидать месяцы или даже годы. Когда же эти продукты наконец выпускаются, часто они ориентированы на уже устаревшее оборудование; также обычно они достаточно дороги и предлагают лишь минимальный набор функций.

Такие длительные циклы выпуска просто недопустимы в современном мире средств связи для бизнеса. Для Интернета распространение новых идей может быть вопросом нескольких недель, и их внедрение занимает очень короткий промежуток времени. Все остальные технологии должны адаптироваться к этим изменениям, это касается и систем связи.

Разработка с открытым исходным кодом по сути своей обладает лучшей способностью приспосабливаться к быстрым технологическим изменениям, что обеспечивает ей гигантское конкурентное преимущество.

Грандиозный крах телефонной отрасли, возможно, в большой мере был обусловлен ее неспособностью меняться. Вероятно, эта неспособность является причиной столь медленного восстановления. Но теперь выбора нет: меняйся или умри. Направляемые сообществами разработчиков открытые технологии, такие как Asterisk, позаботятся об этом.

## Нежелание расстаться с прошлым и раскрыть объятия будущему

Традиционные телекоммуникационные компании утратили связь со своими клиентами. Необходимость введения дополнительных возможностей, кроме базовых функций телефона, отчетливо ясна, но идея того, что именно пользователь должен выбирать эту функциональность, еще не нашла понимания.

Сегодня люди имеют практически неограниченную гибкость во всех других формах связи. Они просто не способны понять, почему услуги телефонной связи не могут быть настолько гибкими, как того обещает данная отрасль в течение уже столь долгих лет. Идея гибкости не близка индустрии связи и, вероятно, не станет таковой до тех пор, пока продукты с открытым исходным кодом, такие как Asterisk, не изменят сути этой отрасли. Это революция, подобная той, начало которой намеренно положили Linux и Интернет более 10 лет назад (и невольно начала IBM, создав ПК за 15 лет до этого). В чем заключается эта революция? В выводе на свободный рынок оборудования и программного обеспечения для телефонии и обеспечении возможности распространения систем телефонной связи, предназначенных для различных потребностей.

# Смена взглядов и понятий

В своей статье «Open Source Paradigm Shift» (*http://tim.oreilly.com/articles/paradigmshift_0504.html*) Тим О'Рейлли (Tim O'Reilly) говорит о смене взглядов и понятий, которая произошла в сфере разработки технологий (это касается как оборудования, так и ПО)[1]. О'Рейлли выделяет три направления: *превращение ПО в товар, сотрудничество, обеспечиваемое сетевыми средствами*, и *возможность гибкой настройки ПО (ПО как услуга)*. Эти три концепции дают основания полагать, что время систем телефонной связи с открытым исходным кодом пришло.

# Перспектива телефонии с открытым исходным кодом

*Все лучшие программы начинаются с решения проблем их автора.*

– Эрик С. Раймонд «Собор и базар»

В своей книге «Собор и базар» Эрик С. Раймонд объясняет, что «при достаточном количестве глаз все дефекты всплывают на поверхность». Причина, по которой разработка ПО с открытым исходным кодом обеспечивает такое устойчивое качество, проста: недостаткам просто невозможно укрыться.

## Проблема, которую решает Asterisk

В эпоху разработки собственных баз данных и веб-сайтов люди не только устали постоянно слышать, что их система телефонной связи «не может обеспечить этого», они просто открыто не верят в это. Изощренные потребности пользователей в сочетании с ограничениями технологии привели к рождению особого типа творчества: специалисты связи, как участники соревнования в передаче «Супервойны на свалке», пытаются создать функциональные устройства из груды плохо сочетающихся компонентов.

Методология разработки узкоспециализированной системы телефонной связи обусловливает наличие в ней множества функций и то, что их количество будет во многом определять цену. Производители обещают, что их продукты предоставят сотни функций, и никого волнует, что вам нужны лишь пять из них. Но хуже всего, что в этой системе может не быть одной функции, без которой вам действительно никак

[1] Следующий раздел во многом является интерпретацией статьи О'Рейлли. Чтобы проникнуть в суть этих идей, настоятельно рекомендуем прочитать оригинальную статью.

не обойтись, и это существенно уменьшит ценность такой системы, потому что она не сможет полностью удовлетворить ваши нужды.

Тот факт, что пользователю могут быть необходимы лишь 5 из 500 функций, не учитывается, и желание пользователя иметь 5 недоступных функций, отвечающих требованиям его деятельности, отвергается как необоснованное[1]. Пока гибкость не станет нормой, телекоммуникация останется в прошлом веке, несмотря на все технологии VoIP в мире.

Asterisk решает именно эту проблему, и решает ее так, как могут немногие системы телефонной связи. Это чрезвычайно разрушительная технология, в большей мере потому, что она основывается на принципах, находящих свое подтверждение снова и снова: «мир с закрытым исходным кодом не может выиграть эволюционную гонку у сообществ, придерживающихся стратегии открытого исходного кода, которые имеют возможность вложить в решение проблемы на порядки больше времени работы квалифицированных специалистов»[2].

## Открытая архитектура

Одним из камней преткновения традиционных систем связи было откровенное нежелание сотрудничать друг с другом. Телекоммуникационные гиганты существуют более ста лет. Принцип закрытых узкоспециализированных систем настолько проник в их культуру, что даже попытки соответствовать стандартам подорваны их желанием обогнать конкурентов, добавив хотя бы одну такую функцию, которую больше никто не поддерживает. Чтобы увидеть пример подобного мышления, достаточно взглянуть на VoIP-продукты, предлагаемые сегодня телекоммуникационной отраслью. Несмотря на заявленное соблюдение стандартов, идея о том, чтобы действительно предоставить возможность подключения телефона Cisco к коммутатору Nortel или интеграции системы голосовой почты Avaya через протокол IP с офисной АТС Siemens, даже не обсуждается.

В компьютерной отрасли все иначе. Двадцать лет назад при покупке IBM-сервера для взаимодействия с ним необходимы были IBM-сеть и IBM-терминалы. Сейчас IBM-сервер, скорее всего, сможет соединять-

---

[1] С точки зрения отрасли с закрытыми исходными кодами такая позиция понятна. В своей книге «The Mythical Man-Month: Essays on Software Engineering» Фред Брукс (Fred Brooks) (издательство Addison-Wesley) говорит, что «сложность и затраты на обмен информацией в рамках проекта возрастают пропорционально квадрату количества разработчиков, тогда как объем выполняемой работы увеличивается линейно». Без привлечения к разработке сообщества создаваемые продукты в лучшем случае будут немного большим, чем просто расширенная и улучшенная версия их предшественников, а в худшем – просто собранием патчей.

[2] Эрик С. Раймонд «Собор и базар».

ся с терминалами Dell по сети Cisco (и работать под управлением ОС Linux, кроме всего прочего). Можно привести массу подобных примеров. Если бы любая из этих компаний заявила, что мы можем использовать их продукты только с тем, на что они нам укажут, ей пришлось бы уйти с рынка.

Телекоммуникационная отрасль переживает такие же изменения, но не торопится принять их. Asterisk, с другой стороны, очень спешит не только принять изменения, но активно использовать их.

IP-телефоны Cisco, Nortel, Avaya и Polycom (и это далеко не полный список) были успешно подключены к системам Asterisk. Сегодня в мире нет другой офисной АТС, которая могла бы похвастать этим. Ни одной.

*Мощь Asterisk – в открытости.*

## Соответствие стандартам

За последние несколько лет стало очевидным: стандарты меняются настолько стремительно, что не отставать от них поможет только умение быстро реагировать на возникающие в технологии новые направления. Asterisk, в силу того что является системой с открытым исходным кодом, создаваемой сообществом разработчиков, идеально подходит для такого типа быстрой разработки, который обеспечит соответствие столь быстро изменяющимся стандартам.

Asterisk не ориентируется на анализ рентабельности или исследования рынка. Она развивается в ответ на все, что, по мнению сообщества, является интересным или необходимым.

## Мгновенный ответ на новые технологии

После посещения первого мероприятия, посвященного тестированию совместимости SIP (SIP Interoperability Test, SIPIT), Марк Спенсер за несколько дней создал для Asterisk элементарный, но рабочий блок, обеспечивающий возможность взаимодействия с библиотекой SIP Stack. Это было еще до того, как SIP стал предпочтительным протоколом в мире VoIP, но Марк разглядел его ценность и потенциал и обеспечил готовность Asterisk к его использованию.

Такое умение предвидеть и гибкость типичны в сообществе разработчиков продуктов с открытым исходным кодом (и очень нетипичны для больших корпораций).

## Полное энтузиазма сообщество

Каждый день в рассылку Asterisk-users (пользователи Asterisk) приходит множество электронных писем. У этой рассылки более 10 000 подписчиков. Такая поддержка сообщества является просто неслыханной для мира узкоспециализированных телекоммуникаций, тогда как в мире открытого исходного кода это норма. Ожидалось, что первое ме-

роприятие AstriCon привлечет внимание 100 человек, пришло около 500 (желающих было намного больше, но они просто не смогли присутствовать). Такая поддержка сообщества, несомненно, гарантирует успех продукта с открытым исходным кодом.

## Что уже возможно

Итак, что можно сделать, используя Asterisk? Давайте рассмотрим кое-что из того, что уже предлагается.

### Шлюз для традиционной офисной АТС

Asterisk может быть фантастическим мостом от старой офисной АТС в будущее. Ее можно разместить перед АТС как шлюз (и выводить пользователей за границы офисной АТС при необходимости) или за АТС как периферийный сервер приложений. Можно даже сделать и то и другое одновременно, как показано на рис. 15.1.

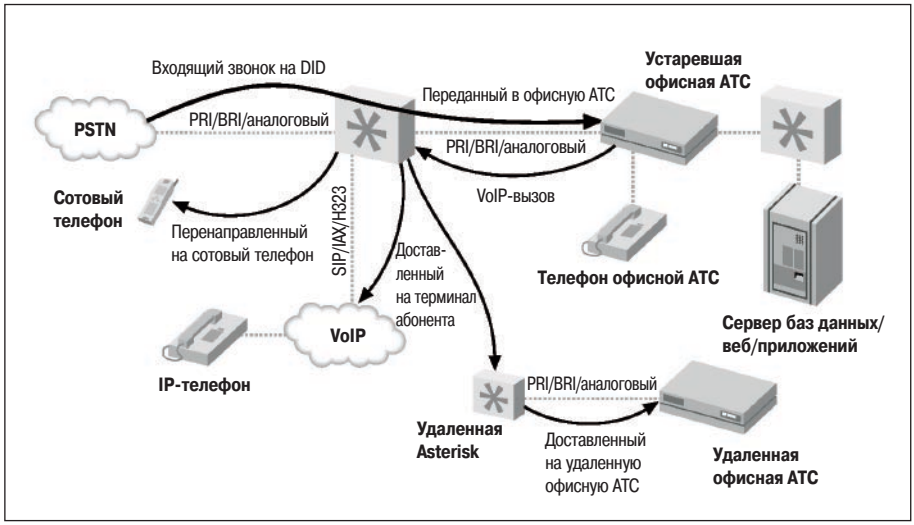



***Рис. 15.1.*** *Asterisk в качестве шлюза офисной АТС*

Вот некоторые функции из тех, что можно реализовать:

*Сохранить старую офисную АТС, но перейти на IP*

Компании, потратившие в последние несколько лет огромные суммы на приобретение узкоспециализированного оборудования офисных АТС, хотят вырваться из тюрьмы ограничений, но не могут смириться с мыслью о том, что придется избавиться от своего оборудования, работающего неэффективно. Нет проблем. Asterisk может решить любые задачи, от замены системы голосовой почты до предоставления возможности введения пользователей, поддерживающих IP, сверх номинальной емкости системы.

*Найди меня, следуй за мной*

Предоставьте офисной АТС список номеров, по которым можно с вами связаться, – и она будет звонить по ним при поступлении вызова на ваш DID (Direct Inward Dialing, то есть телефонный номер). Рис. 15.2 иллюстрирует эту технологию.

*Звонки по VoIP*

Если можно установить соединение по телефонной линии между офисной АТС Asterisk и старой офисной АТС, Asterisk может обеспечить доступ к сервисам VoIP, тогда как старая офисная АТС продолжает соединяться с внешним миром, как обычно. При использовании в качестве шлюза Asterisk просто надо эмулировать функции PSTN – и старая офисная АТС даже не будет знать, что что-то изменилось. На рис. 15.3 показано, как можно использовать Asterisk для обеспечения поддержки VoIP в устаревшей офисной АТС.

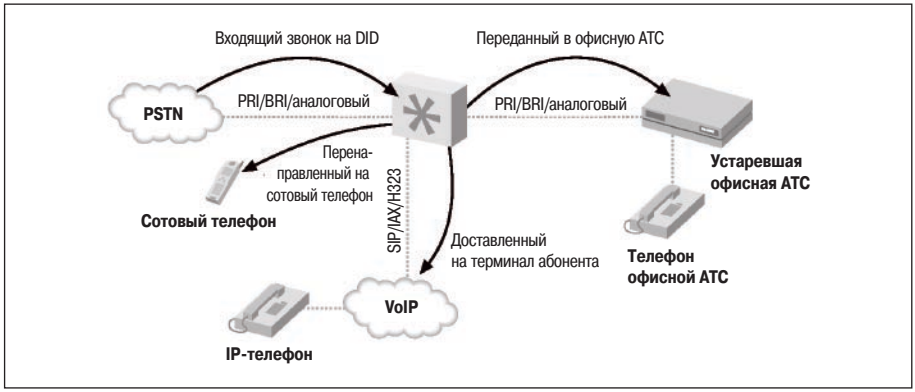



***Рис. 15.2.*** *Найди меня, следуй за мной*

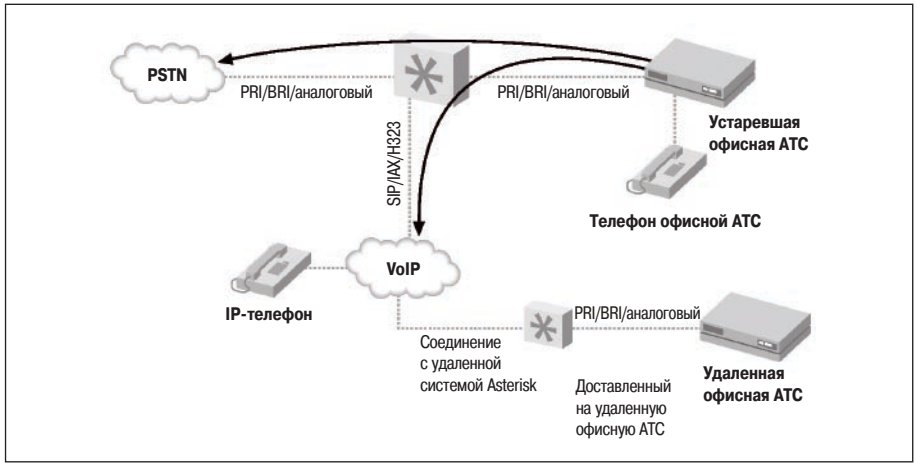



***Рис. 15.3.*** *Поддержка VoIP в устаревшей офисной АТС*

## Доступный IVR

Многие путают интерактивный автоответчик (Interactive Voice Response, IVR) с обычным автоответчиком (Automated Attendant, AA). Поскольку изначально IVR использовался как автоответчик, это неудивительно. Тем не менее понятие IVR в телефонии намного шире, чем просто автоответчик. AA является, так сказать, средством для перенаправления вызывающего абонента на добавочный номер и встроен в большинство узкоспециализированных систем голосовой почты. Но функционал IVR может быть намного бóльшим.

Системы IVR обычно не просто дороги по цене, их конфигурирование очень трудоемко. Для специальной системы IVR обычно требуется возможность соединения с внешней базой данных или приложением. Asterisk – пожалуй, идеальный IVR, поскольку в ней заложена возможность соединения с базами данных и приложениями на самом глубоком уровне.

Вот несколько примеров относительно простых IVR, для создания которых можно использовать систему Asterisk:

*Получение прогноза погоды*

Используя Интернет, можно получать прогноз погоды в текстовом виде со всего мира сотнями способов. Если взять эти отчеты и обработать специальным синтаксическим анализатором (сценарий Perl, вероятно, мог бы с этим справиться), эту информацию можно сделать доступной для диалплана. Фонотека Asterisk уже имеет все необходимые голосовые сообщения, поэтому не составит труда создать интерактивное меню для воспроизведения текущих метеорологических прогнозов для любой точки земного шара.

*Математические программы*

Эд Гай (Ed Guy, архитектор сети FWD (Free World Dialup – бесплатные звонки по всему миру) Пулвера) представил на конференции AstriCon 2004 небольшую математическую программу, которую создал для своей дочери. На ее написание ушло не более часа. Она представляла ряд математических вопросов, ответы на которые вводились посредством номеронабирателя телефона. После ответа на все вопросы система сообщала оценку. Реализация такого предельно простого Asterisk-приложения в любой офисной АТС на закрытой платформе стоила бы десятки тысяч долларов, если вообще могла бы быть выполнена. Подробнее об этом рассказывается в главе 9. Как это часто бывает, то, что просто для Asterisk, практически невозможно или чрезвычайно дорого в любой другой системе IVR.

*Распределенные IVR*

Цена узкоспециализированной системы IVR такова, что, если компания, имеющая множество торговых точек, хочет обеспечить функциональность IVR, для обработки транзакций она вынуждена перенаправлять абонентов на центральный сервер. С Asterisk становит-

ся возможным распространить приложение на все серверы и, таким образом, обрабатывать запросы локально. Буквально тысячи маленьких систем Asterisk, развернутых в торговых точках по всему миру, могли бы обеспечивать функциональность IVR так, как не способна ни одна другая система. И не нужно никаких удаленных переадресаций на центральный сервер IVR или используемых специально для этой цели гигантских магистральных каналов – больше возможностей за меньшие деньги.

Это три довольно простых примера того, на что потенциально способна Asterisk.

## Конференц-залы

Эта маленькая возможность в конце концов станет одной из самых потрясающих функций Asterisk. В сообществе разработчиков Asterisk конференц-залы используются для разных целей все чаще:

- Небольшим компаниям необходим простой способ обеспечить возможность деловым партнерам собраться и обсудить некоторые вопросы.
- Группы сбыта встречаются раз в неделю, при этом каждый должен иметь возможность позвонить оттуда, где он находится в данное время.
- Группы разработки назначают единое место и время для передачи друг другу информации о внесенных изменениях.

## Бытовая автоматизация

Asterisk по-прежнему во многом является инструментом для суперфанатов и поэтому вряд ли может использоваться в обычном доме, но при наличии у вас незаурядных знаний по Linux и Asterisk становится возможным следующее.

*Контроль за детьми*

Родители, желающие контролировать няню (или детей, оставшихся дома одних), могут звонить в контекст добавочного номера, защищенный паролем. После аутентификации будет устанавливаться двусторонняя аудиосвязь со всеми IP-телефонами в доме – это позволит маме и папе слышать, что там происходит. Жутко? Да, но интересная идея тем не менее.

*Блокировка телефонов*

Уходите на ночь? Не хотите, чтобы няня висела на телефоне? Нет проблем! Небольшое изменение диалплана – и можно звонить только на телефоны экстренного вызова, ваш мобильный телефон и в пиццерию. Попытки позвонить на любой другой телефон приведут к воспроизведению записи «Мы платим вам за присмотр за нашими детьми, а не за ваши личные звонки».

Очень коварно, правда?

*Управление системой сигнализации*

Пребывая на отдыхе, получаете звонок о том, что ваша мама хочет взять у вас какую-то кухонную утварь. Она забыла свой ключ и стоит под дверью вашего дома, дрожа от холода. Проще простого. Звонок в свою систему Asterisk, передача короткой строки цифр в специально созданный для этой цели контекст – и ваша система сигнализации получает указание отключить сигнализацию на 15 минут. Посоветуйте маме поторопиться забрать, что ей нужно, и покинуть помещение побыстрее, или нагрянет полиция!

*Управление звонками детей*

Как насчет того, чтобы установить временное ограничение на звонки своих детей? Чтобы использовать телефон, они должны ввести свой код доступа. Дополнительные минуты можно зарабатывать, выполняя работу по дому, за хорошие оценки, выставив за дверь этого надоедливого бездельника с ужасной прической, – идея ясна? Как только они потратили все свои минуты... щелк... телефоном можете пользоваться только вы.

С помощью Caller ID (ID звонящего) можно управлять и входящими звонками. «Донни, это отец Сьюзи. Она больше не хочет тебя видеть, поскольку решила немного поднять планку своих стандартов. И еще, тебе не мешало бы подстричься».

# Будущее Asterisk

Мы полюбили Интернет и за богатство содержимого, и за дешевизну, и, что, вероятно, еще более важно, за то, что он позволяет нам самим решать, как общаться. По мере расширения его возможностей по переносу различных все более богатых форм медиа-данных мы будем использовать его чаще и чаще. Как только Интернет обеспечит доставку голоса с качеством, сравнимым с тем (или лучшим), что предоставляет PSTN, телефонной компании придется менять сферу деятельности. PSTN прекратит свое существование и станет не более чем еще одним протоколом связи, предоставляемым Интернетом. Как это произошло с большей частью Интернета, технологии с открытым исходным кодом обеспечат такое превращение.

## Обработка речи

Мечта о том, чтобы наши технические изобретения разговаривали с нами, старше, чем сам телефон. Каждая новая ступень в развитии технологии порождала новую волну активных экспериментов. Как правило, результаты не соответствовали ожиданиям, вероятно, потому, что, как только машина начинает говорить то, что кажется разумным, большинство людей действительно начинают верить в ее разумность.

Люди, создающие программы и обслуживающие компьютеры, осознают ограниченность их возможностей и, таким образом, допускают не-

которые недостатки таких систем. Все остальные просто хотят, чтобы их компьютеры и программное обеспечение работали. Количество умственных усилий, которые пользователь должен приложить при взаимодействии с компьютером, обратно пропорционально объему умственных затрат, вложенных командой разработки. За простыми интерфейсами стоят сложные проектные решения.

Следовательно, требуется разработать систему, которая предугадывала бы наиболее распространенные желания пользователей, а также могла искусно справляться с неожиданными задачами.

## Festival

Сервер речевого воспроизведения текста Festival преобразует текст в проговариваемые слова. Хотя это довольно забавная штука, с которой очень весело играть, но необходимо решить еще очень много проблем.

Очевидное и ценное применение преобразования текста в речь для Asterisk – чтение вслух электронной почты. Конечно, если учесть грамматические, пунктуационные и орфографические ошибки, которыми изобилуют сообщения электронной почты в наши дни, пожалуй, можно представить, насколько это сложная задача.

Остается только надеяться, что появление возможности речевого воспроизведения текста вдохновит новое поколение обращать больше внимания на грамотность написания. Видеть орфографические и пунктуационные ошибки на экране достаточно неприятно; но чтобы без слез выслушивать, как компьютер проговаривает это, необходимо иметь просто олимпийское спокойствие, чем могут похвастать очень немногие.

## Распознавание речи

Если преобразование текста в речь – это «космическая технология», то распознавание речи – это научная фантастика.

На самом деле распознавание речи возможно с довольно высоким качеством, но, к сожалению, обычно для этого необходимо обеспечить соответствующие условия, и это совсем не те условия, которые предлагает телефонная сеть. Даже идеальное PSTN-соединение обеспечивает минимально допустимый предел качества для точного распознавания речи. Добавьте сюда VoIP-соединения со сжатием или потерей качества или сотовый телефон – и вы обнаружите намного больше ограничений, чем преимуществ.

Asterisk сейчас имеет готовый полностью речевой API, так что внешние компании (или даже проекты с открытым исходным кодом) могут подключать свои механизмы распознавания речи к Asterisk. Одна компания, сделавшая это, – LumenVox. Используя ее механизм распознавания речи вместе с Asterisk, можно за рекордно короткое время создать управляемые голосом меню и системы IVR! Больше информации можно найти по адресу *http://www.lumenvox.com*.

## Высокая точность воспроизведения голоса

По мере того как становятся доступны все более широкие полосы пропускания, все труднее понимать, почему до сих пор используются кодеки, обеспечивающие такую низкую точность воспроизведения. Многие даже не представляют, что Skype использует более высокую точность воспроизведения, чем телефон; главным образом поэтому Skype имеет репутацию системы с хорошим звучанием.

Если вы когда-нибудь звонили на CNN, разве не приятно было услышать сладкозвучный голос Джеймса Эрла Джонса, говорящий: «Это CNN», – вместо какой-то электронной записи с металлическим звуком. И если вы полагаете, что голос Эллисон Смит[1] звучит в телефоне хорошо, вам стоит поговорить с ней лично!

В будущем мы ожидаем – и добьемся этого, – что наше оборудование связи будет обеспечивать высокоточное воспроизведение голоса.

Asterisk начиная с версии 1.4 предоставляет ограниченную поддержку кодека G.722. По мере того как все больше производителей оборудования начнут встраивать поддержку высокоточного воспроизведения голоса в свои устройства для VoIP, будет расширяться его поддержка в Asterisk, что сделает качество звонков выше, чем в PSTN.

## Видео

Хотя данная книга преимущественно посвящена работе с аудиоданными, Asterisk обеспечивает разнообразную поддержку и для видеоданных. Однако эта поддержка неполная. Проблема не столько в функциональности, сколько в пропускной способности и вычислительной мощности. Что еще более важно – сообщество еще не видит большой необходимости уделять данному вопросу повышенное внимание.

### Проблема видеоконференц-связи

Идея видеоконференц-связи витала в воздухе с момента появления электронно-лучевой трубки. Телекоммуникационная отрасль десятилетиями обещала предоставить устройство видеоконференц-связи в каждый дом.

Как и в случае со многими другими технологиями связи, если в вашем доме есть устройство видеоконференц-связи, скорее всего, оно работает по Интернету с использованием простой недорогой веб-камеры. Однако, похоже, люди считают видеосвязь чем-то условным. Да, вы можете видеть человека, с которым разговариваете, но чего-то не хватает.

---

[1] Эллисон Смит – «голос Asterisk»; это она озвучила все голосовые сообщения системы. Чтобы записать собственные сообщения голосом Эллисон, просто посетите веб-сайт *http://thevoice.digium.com*.

### Почему мы любим видеконференц-связь

Видеконференц-связь предлагает более полноценное общение, чем телефон. Вы получаете возможность не просто слышать обезличенный голос, но и улавливать все нюансы речи, передаваемые лишь при непосредственном общении.

### Почему видеоконференц-связь, возможно, никогда полностью не заменит аудиосвязь

Необходимо решить несколько проблем, и не все они находятся в технической плоскости.

Смотрите: используя обычный телефон, люди, работающие на дому, могут вести деловые переговоры в нижнем белье, положив ноги на стол, с чашкой кофе в руке. Аналогичная видеосессия потребовала бы получасовой подготовки и приведения себя в порядок и не могла бы вестись с кухни, террасы или... ну, вы поняли.

Также все возможности визуального контакта по видеосвязи ничего не стоят, если фокальные точки собеседников не располагаются на одной линии с камерами. Если вы будете смотреть в камеру, ваша аудитория будет видеть, что вы смотрите на них, но вы не будете видеть их. Если вы будете смотреть на свой экран, чтобы увидеть того, с кем разговариваете, камера будет показывать, что вы смотрите куда-то в сторону, не на аудиторию. Это создает впечатление отчужденности. Возможно, если бы видеофон мог быть спроектирован, как TelePromptR, где камера располагалась за экраном, не создавалось бы впечатление такой неестественности. В данных условиях не хватает чего-то с точки зрения психологии. Видео прекращает быть диковинкой.

## Беспроводная связь

Поскольку Asterisk полностью поддерживает VoIP, поддерживается и функция беспроводной связи.

### Wi-Fi

Wi-Fi станет решением для обеспечения мобильности в рамках офиса для VoIP-телефонов. Эта технология уже достаточно зрелая. Самая большая помеха – стоимость телефонных трубок. Но эта ситуация обещает улучшиться, поскольку рост конкуренции по всему миру приводит к снижению цен.

### Wi-MAX

Мы так смело делаем свои многочисленные предсказания, поэтому не побоимся заявить, что технология Wi-MAX стала началом конца традиционных сетей сотовой телефонной связи.

При наличии беспроводного доступа в Интернет в большинстве населенных пунктов какой смысл в дорогой сотовой связи?

## Универсальная система передачи и обработки сообщений

Это рекламировалось телефонной индустрией годами, но внедрение проходит намного медленнее, чем предсказывалось.

Универсальная система передачи и обработки сообщений – это концепция совмещения систем голосовых и текстовых сообщений. Теперь их не надо объединять искусственно, поскольку Asterisk уже обрабатывает их одинаково.

Даже просто взглянув на этот термин и увидев слова «универсальная» и «передача и обработка сообщений», можно понять, что интеграция электронной и голосовой почты должна быть только началом. Универсальная система передачи и обработки сообщений, чтобы оправдать свое название, должна делать намного большее.

Возможно, необходимо определить «передачу и обработку сообщений» как связь, которая происходит не в режиме реального времени. Иначе говоря, при отправке сообщения вы предполагаете, что ответ может поступить через мгновение, несколько минут, часов или даже дней. Вы формулируете то, что хотите сказать, и ожидается, что ваша аудитория даст ответ.

Сравните это с разговором, происходящим в реальном времени. Говоря по телефону, вы ожидаете ответа не более нескольких секунд.

В 2002 году Тим О'Рейлли сделал доклад под названием «Watching the Alpha Geeks: OS X and the Next Big Thing» (*http://www.macdevcenter.com/pub/a/mac/2002/05/14/oreilly_wwdc_keynote.html*), в котором рассказывал о передаче IRC через механизм речевого воспроизведения текста. Можно представить и обратное: позволяя нам присоединяться к IRC или системе мгновенных сообщений по Wi-Fi-телефону, наша Asterisk-АТС обеспечивает преобразование речи в текст и опять в речь.

## Равноправный информационный обмен

Поскольку сети-монополисты, в числе которых PSTN, сдали позиции таким поддерживаемым сообществами пользователей сетям, как Интернет, придет время, когда понадобится обеспечить возможность их соединения. Хотя традиционные поставщики услуг предпочли бы, чтобы существующая модель была перенесена в новую систему, скорее всего, телефонные звонки станут не более чем еще одним интернет-приложением.

Но проблема остается: как сохранить телефонный план нумерации, который уже стал настолько привычным и удобным?

### E.164

ITU определил план нумерации в своей спецификации E.164. Если вы когда-нибудь использовали телефон для звонков по PSTN, можете

с уверенностью заявлять, что знакомы с концепцией нумерации по E.164. До появления общедоступного VoIP никто не заботился о E.164, кроме телефонных компаний. В этом просто не было необходимости.

Теперь, когда звонки путешествуют из PSTN в Интернет и бог знает куда еще, на E.164 необходимо обратить внимание.

### ENUM

В ответ на эту проблему IETF профинансировал рабочую группу по разработке протокола отображения телефонных номеров Telephone Number Mapping (ENUM), целью которой было объединение номеров E.164 со службой доменных имен (Domain Name System, DNS).

Идея ENUM разумна, но для достижения успеха необходима поддержка со стороны телекоммуникационной отрасли. Однако желание сотрудничать никогда не входило в число добродетелей этой отрасли, поэтому проект ENUM потерпел неудачу.

### e164.org

Ребята с *e164.org* пытаются обеспечить успех ENUM. Можно зарегистрироваться на этом сайте, зарегистрировать свой номер телефона и сообщить системе альтернативные методы, по которым возможно связаться с вами. Это означает, что кто-то, кому известен ваш номер телефона, сможет позвонить вам по VoIP, поскольку DNS-зона *e164.org* предоставит информацию об IP-адресе и протоколе, необходимую, чтобы связаться с вами.

По мере того как все больше и больше людей будут предоставлять сведения для возможности подключения по VoIP, все меньшее и меньшее число звонков будет осуществляться по PSTN.

### DUNDi

Распределенный универсальный каталог номеров (DUNDi) – это открытый протокол маршрутизации, разработанный для обслуживания динамических таблиц маршрутизации, которые предназначены для соединения между совместимыми системами (больше информации можно найти в главе 14). Хотя в настоящее время Asterisk является единственной офисной АТС, поддерживающей DUNDi, открытость этого стандарта позволяет реализовать его кому угодно.

DUNDi имеет огромный потенциал, но он еще находится в начальной стадии развития, не стоит упускать это из виду.

## Проблемы

Как и любое стоящее начинание, Asterisk столкнется с проблемами. Давайте кратко рассмотрим, какого рода трудности это могут быть.

## Слишком много изменений, слишком мало стандартов

В наши дни Интернет меняется так стремительно и предлагает настолько разнообразную информацию, что даже самый внимательный фанат не сможет уследить за всем. Так и должно быть, но это также означает, что невероятная скорость смены используемых технологий является обязательным условием поддержания любой системы связи в соответствии с текущими требованиями.

## Спам по VoIP

Да, он грядет. Всегда были люди, которые верили, что они имеют право доставлять неудобства и беспокоить других в своей погоне за наживой. Средства борьбы с этим находятся на стадии разработки, но только время покажет, насколько эффективными они будут.

## Страх, неопределенность и сомнение

Отрасль находится на стадии перехода от игнорирования к осмеянию. Если Ганди прав, можно ожидать скорой войны.

По мере все более стремительного уменьшения своих доходов, вызванного распространением телефонии с открытым исходным кодом, традиционные игроки на рынке телекоммуникаций, безусловно, развяжут компанию запугивания в надежде помешать революционным изменениям.

## Искусственное создание узких мест

Ширятся слухи, что крупные операторы сети телекоммуникаций начнут вмешиваться в трафик VoIP, помечая и отдавая предпочтение трафику своих VoIP-сервисов, предоставляемых за дополнительную плату, и, самое худшее, выявляя и устраняя любой VoIP-трафик, сформированный сервисами, которые не были ими санкционированы.

Кое-что из этого уже происходит: поставщики сервисов блокируют в своих сетях прохождение определенного трафика, объясняя это выполнением неких общественных функций (например, блокирование популярных сервисов общего доступа к файлам для защиты нас от пиратства). В США Федеральная комиссия по связи (ФКС) заняла четкую позицию по данному вопросу и оштрафовала компании, замеченные в таких действиях. В остальном мире правоохранительные органы не всегда так благосклонны к VoIP.

Очевидно лишь то, что сообщество и Сеть найдут способы обойти преграды, как они всегда это делали.

## Нормативно-правовые войны

Недавно вышедший в отставку руководитель Федеральной комиссии по связи США Майкл Пауэлл (Michael Powell) сделал подарок, который, возможно, изменит путь развития революции VoIP. Вместо того

чтобы делать попытки регулирования VoIP как телекоммуникационного сервиса, он поддержал идею о том, что VoIP представляет совершенно новый способ связи и требует собственного правового поля, в котором будет развиваться.

VoIP станет регламентируемым, но не везде как сервис телефонной связи. Вот некоторые из возможных норм:

*Наличие информации для экстренных служб*

Одна из характеристик традиционной сети PSTN – она не меняет своего местоположения. Это очень полезно для экстренных служб, поскольку они могут определить местоположение вызывающего абонента, определив адрес телефонной сети, из которой поступил звонок. Распространение сотовых телефонов сильно усложнило ситуацию, поскольку сотовый телефон не имеет определенного адреса. Он может подключиться к любой сети и зарегистрироваться на любом сервере. Если телефон не предоставляет информации о своем физическом местоположении, вызов экстренной службы с него не обеспечит никаких данных о том, где находится звонящий. VoIP создает аналогичные проблемы.

*Отслеживание звонков для правоохранительных органов*

Правоохранительные органы всегда имели возможность перехватывать телефонные разговоры по традиционным коммутируемым телефонным линиям. Тогда как приняты законы, призванные добиться того же в Сети, техническая сторона обеспечения такой функциональности, вероятно, никогда не будет решена полностью. Люди очень трепетно относятся к своей частной жизни, и чем активнее власти посягают на нее, тем больше усилий будет направлено на ее сохранение.

*Меры борьбы с монополиями*

Эти меры уже применяются в США: взимание штрафов с сетевых провайдеров, которые пытаются фильтровать трафик на основании содержимого.

Что касается регулирования, здесь Asterisk выступает в роли и ангела, и беса одновременно: ангела – потому, что помогает бедным, а беса – потому, что обеспечивает богатейшие возможности хакерам и спамерам как никто другой. Регулирование телефонии с открытым исходным кодом частично может определяться уровнем саморегулируемости сообщества разработчиков. Такие концепции, как DUNDi, которые включают процессы, направленные против распространения спама, являются превосходным началом. С другой стороны, такие идеи, как подделка идентификатора вызывающего абонента, – готовая площадка для злоупотреблений.

## Качество и класс предоставляемых услуг передачи данных

Реальность такова, что Интернет на базе TCP/IP обеспечивает негарантированное обслуживание. Из-за этого пока невозможно сказать, как

стремительно растущий VoIP-трафик реального времени повлияет на общую производительность Сети. В настоящее время превышение имеющейся полосы пропускания магистральной сети таково, что обеспечивает приемлемое качество даже в условиях негарантированного обслуживания. Однако мы не раз убеждались, что какая бы большая полоса пропускания ни была предоставлена, мы всегда найдем ей применение. DSL-соединения 1 Мб, немыслимого пять лет назад, сегодня едва хватает.

Наверное, следствие закона Мура[1] будет применимо и к пропускной способности Сети. Вопрос качества услуг может стать неактуальным по причине способности Сети предоставлять необходимую производительность без какой-либо специальной обработки. Организации, требующие более высоких уровней надежности, могут предпочесть отдельно оплачивать более качественные услуги. Возможно, эру поминутной оплаты междугородных соединений сменит эпоха помиллисекундной оплаты за связь с гарантированной низкой задержкой или за каждую долю процентов сокращения потери пакетов. Сервисы, предоставляемые за дополнительную плату, предлагают уровень надежности с пятью девятками[2], что традиционные телефонные компании всегда преподносят как аргумент в свою пользу против VoIP.

## Сложность

Открытые системы требуют новых подходов к проектированию решения. Дешевизна оборудования и ПО еще не означает, что таким же дешевым будет и решение. Asterisk не является стандартным ПО, готовым к использованию; его надо спроектировать и собрать, а затем и обслуживать. Поскольку базовое ПО бесплатно и цены на оборудование будут формироваться на основании конъюнктуры рынка, справедливости ради следует отметить, что для высокоспециализированной системы стоимость настройки будет составлять значительную часть цены решения. Во многих случаях из-за высокой сложности и гибкости конфигурации эта составляющая больше, чем она была бы для традиционной офисной АТС.

---

[1] Гордон Мур в 1965 году написал статью, в которой предсказал удвоение количества транзисторов в процессоре каждые несколько лет.

[2] Здесь речь идет о 99,999%, что преподносится как степень надежности, предоставляемая традиционными телефонными сетями. «Пять девяток» означает, что бездействие сервиса в год не может превышать 5 мин 15 с. Многие верят, что VoIP должен будет достичь такого уровня надежности, прежде чем можно будет ожидать, что он полностью заменит PSTN. Другие верят, что PSTN даже не приближается к надежности с «пятью девятками». Мы уверены, что эта величина могла бы быть прекрасным показателем для описания высокой надежности, но отделы сбыта слишком злоупотребляют им.

Обычно основное правило такое: если это может быть сделано в диалплане, проектирование системы займет примерно столько же времени, как и проектирование любой традиционной офисной АТС с аналогичными функциями. Для всего, что сверх этого, оценить время, необходимое для построения системы, можно только исходя из опыта.

Здесь еще многому надо учиться.

## Возможности

Телефония с открытым исходным кодом создает неограниченные возможности. Вот некоторые из наиболее привлекательных.

### Частные телекоммуникационные сети, выполненные на заказ

Кто-то скажет, что главное – цена, но мы верим, что реальное основание для будущего успеха Asterisk в том, что она позволяет создавать систему телефонной связи так же, как мы создавали бы веб-сайт: с возможностью полной настройки каждого аспекта системы. Пользователи мечтали об этом годами. Только Asterisk может обеспечить это.

### Доступность

Каждый может сделать свой вклад в будущее связи. Теперь любой владелец старенького ПК стоимостью 200 долларов в состоянии создать систему связи с настолько развитой логикой, что она может конкурировать с самыми дорогими узкоспециализированными системами. Даже если оборудование не готово к производственной эксплуатации, нет такой причины, по которой ПО не было бы таковым. Это одна из причин, по которой закрытые системы ожидают трудные времена конкуренции. То огромное число людей, которые имеют доступ к необходимому оборудованию, не идет ни в какое сравнение с количеством членов закрытого клуба.

### Решения, размещаемые на ресурсах поставщиков услуг, сложность которых сравнима со сложностью корпоративных веб-сайтов

Проектирование офисной АТС всегда было некоторой формой искусства, но до Asterisk искусство состояло в поиске путей обойти ограничения технологии. При наличии технологии с неограниченными возможностями те же творческие подходы теперь помогут полностью реализовать все нужды клиентов. Системы телефонной связи с открытым исходным кодом, такие как Asterisk, сделают это возможным. Разработчики телефонных систем будут плясать от радости, потому что теперь их огромный творческий потенциал реально послужит клиентам, а не будет растрачиваться на обход дефектов.

## Реальная интеграция технологий связи

В конечном счете все надежды на технологии с открытым исходным кодом пойдут прахом, если они не смогут удовлетворить потребность людей в решении проблем. Закрытые отрасли забыли о клиенте и пытаются приспособить пользователя к продукту.

Телефония с открытым исходным кодом ставит телефонную связь в один ряд с другими информационными технологиями. Наконец можно приступить к собственно реализации задачи по интеграции электронной почты, голоса, видео и всего, что только можно представить, по гибким транспортным сетям (как проводным, так и беспроводным) в ответ на нужды пользователя, а не по прихоти монополий.

Добро пожаловать в будущее телекоммуникаций!

# A

# Каналы VoIP

Каналы VoIP в Asterisk представляют средство соединения с протоколами, которые они поддерживают. Для использования протокола необходим конфигурационный файл, где содержатся общие параметры, описывающие, как система обрабатывает протокол, а также специальные параметры для каждого канала (или устройства), которые будут упоминаться в диалплане. В данном приложении подробно рассматриваются конфигурационные файлы протоколов IAX и SIP.

## IAX

Конфигурационный файл IAX (iax.conf) содержит всю конфигурационную информацию, необходимую Asterisk для создания и управления каналами, работающими по протоколу IAX. Разделы файла отделены заголовками, сформированными словом, заключенным в квадратные скобки (`[]`). Имя в скобках будет именем канала за одним исключением: раздел `[general]` не является каналом, это область, где определяются глобальные параметры протокола.

В данном разделе рассматриваются различные общие и специальные настройки канала, определяемые в файле iax.conf. Мы опишем каждый параметр, а затем приведем пример его использования. Некоторые опции могут иметь несколько действительных аргументов. Список этих аргументов, разделенных символом вертикальной черты (|), приводится рядом с опцией. Например, запись `bandwidth=low|medium|high` означает, что параметр `bandwidth` принимает в качестве аргумента одно из значений: `low`, `medium` или `high`.

Комментарии можно вставлять в любом месте файла iax.conf. Текст комментария начинается с точки с запятой (;). Все, что располагается справа от точки с запятой, будет проигнорировано обработчиком Asterisk. Комментарии могут использоваться достаточно активно.

## Общие настройки IAX

Первой незакомментированной строкой в файле iax.conf должен быть заголовок `[general]`. Параметры в этом разделе будут применяться ко всем соединениям, использующим данный протокол, если не определено другое в описании конкретного канала. Поскольку некоторые из этих настроек могут определяться для канала индивидуально, меткой (глобальный) мы обозначили настройки, которые всегда глобальны, а меткой (канал) – те, которые могут задаваться для каждого канала в отдельности. Если параметр задается в разделе `[general]`, его не надо определять для каждого канала; его значение становится значением по умолчанию. Помните, что определение параметра в разделе `[general]` не запрещает задавать ему другое значение для конкретного канала; это просто делает данное значение значением по умолчанию. Также нельзя забывать, что отсутствие задания этих параметров может в некоторых случаях приводить к использованию системных значений по умолчанию.

Вот параметры, которые могут быть сконфигурированы:

`accountcode` (канал)

Код учетной записи может определяться для каждого пользователя. Если задан, этот код учетной записи будет присваиваться записи вызова, если не задан код учетной записи конкретного пользователя. Заданное имя `accountcode` будет использоваться как имя файла в формате CSV в папке /var/log/asterisk/cdr-csv/, где хранятся записи параметров вызовов (Call Detail Records, CDRs) для соединений типа `user/peer/friend`:

```
accountcode=iax-имяпользователя
```

`adsi` (канал)

Поддержка ADSI (Analog Display Services Interface – интерфейс сервисов для аналогового дисплея) может быть активирована только при наличии совместимого с ADSI телекоммуникационного оборудования на стороне клиента (CPE-оборудования):

```
adsi=yes|no
```

`allow` и `disallow` (канал)

Могут быть разрешены или запрещены определенные кодеки, что позволяет разработчику системы задавать перечень используемых кодеков. `allow` и `disallow` также могут быть определены для канала отдельно. Помните, что выражения `allow` в разделе `[general]` будут распространяться на все каналы, для которых не переопределено `disallow=all`. Согласование кодеков ведется в порядке их задания.

Лучшей практикой считается определять `disallow=all`, а затем в выражениях `allow` явно задавать каждый кодек, который вы желаете использовать. Если ничего не задано, предполагается `allow=all`:

```
disallow=all
allow=ulaw
allow=gsm
allow=ilbc
```

`amaflags` (канал)

Система автоматической регистрации сообщений (Automatic Message Accounting, AMA) описана в наборе документов компании Telcordia, зарегистрированных как FR-AMA-1. Эти документы определяют стандартные механизмы формирования и передачи CDR. Можно задать один из четырех флагов AMA, который будет применяться ко всем IAX-соединениям:

```
am flags=default|omit|billing|documentation
```

`authdebug` (глобальный)

Можно свести до минимума объем записи параметров аутентификации, отключив его с помощью `authdebug=no`. Запись параметров аутентификации активирована по умолчанию, если не отключена явно:

```
authdebug=no
```

`autokill` (глобальный)

Чтобы максимально сократить опасность зависания в условиях недоступности хоста, можно задать для параметра `autokill` значение `yes`, тогда любое новое соединение будет разорвано при отсутствии подтверждения `ACK` в течение 2000 мс. (Очевидно, что это не рекомендуется для хостов с большой задержкой.) Вместо `yes` можно указать время ожидания в миллисекундах перед тем, как будет принято решение о недоступности равноправного участника сети. Параметр `autokill` задает время ожидания для всех равноправных участников, работающих по протоколу IAX2, но с помощью команды `qualify` его можно сконфигурировать для каждого равноправного участника в отдельности:

```
autokill=1500
```

`bandwidth` (канал)

`bandwidth` – это сокращенная запись, которая поможет избежать применения `disallow=all` и множества выражений `allow` для задания используемых кодеков. Допустимыми опциями являются:

`high`

Допускаются все медиа-кодеки (G.723.1, GSM, µlaw, alaw, G.726, ADPCM, slinear, LPC10, G.729, Speex, iLBC).

`medium`

Допускаются все кодеки, кроме slinear, µlaw и alaw.

`low`

Допускаются все медиа-кодеки, кроме G.726 и ADPCM.

```
bandwidth=low|medium|high
```

`bindport` и `bindaddr` (глобальные)

Эти необязательные параметры позволяют задавать IP-интерфейс и порт, которые будут принимать IAX-соединения. Если они опущены, будет задан порт 4569, и все IP-адреса системы Asterisk будут принимать входящие IAX-соединения. Если задано несколько адресов привязки, IAX-соединения будет принимать только заданный интерфейс[1]. Использование вместо адреса `0.0.0.0` указывает Asterisk слушать все интерфейсы:

```
bindport=4569
bindaddr=192.168.0.1
```

`codecpriority` (канал)

Опция `codecpriority` определяет, какие параметры при входящем вызове будут иметь преимущество при согласовании кодеков. Если этот параметр задан в разделе `[general]`, выбранные опции будут наследоваться всеми пользовательскими настройками в конфигурационном файле канала; однако их можно задавать в отдельных пользовательских параметрах для более тонкой настройки. Если значения заданы и в `[general]`, и в пользовательских разделах, настройки пользователя переопределяют настройки раздела `[general]`. Если этот параметр не задан, значение по умолчанию – `host`.

К допустимым опциям относятся:

`caller`

Вызывающая сторона имеет приоритет над хостом.

`host`

Хост имеет приоритет над вызывающей стороной.

`disabled`

Предпочтения кодеков не учитываются; это поведение по умолчанию, если не определены предпочтения кодеков.

`reqonly`

Предпочтения кодеков игнорируются, и вызовы принимаются, только если запрашиваемый кодек доступен:

```
codecpriority=caller|host|disabled|reqonly
```

[1] В настоящее время Asterisk работает только с одной опцией `bindaddr`. Если необходимо слушать более одного адреса, придется использовать `0.0.0.0`. Обратите внимание, что Asterisk работает в многосетевой среде, но не с многоадресными интерфейсами. Asterisk выбирает интерфейс, с которого будет отправлен пакет, используя таблицу маршрутизации системы, и в качестве исходного применяется основной адрес этого интерфейса.

`delayreject` (глобальный)

Если по IAX-каналу поступает ошибочный пароль, это обеспечит задержку отправки сообщений отказа `REGREQ` или `AUTHREP`, что поможет защититься от атак с подбором паролей. Время задержки – 1000 мс:

```
delayreject=yes|no
```

`forcejitterbuffer` (канал)

Поскольку Asterisk пытается соединить каналы (конечные точки) напрямую, конечным точкам обычно позволено самостоятельно выполнять буферизацию входящих сообщений, которая будет компенсировать задержки. Однако, если конечные точки имеют слабую реализацию такой буферизации, можно заставить Asterisk выполнять ее всегда, независимо от условий. Чтобы принудительно активировать буферизацию входящих сообщений для компенсации задержки, необходимо задать `forcejitterbuffer=yes`:

```
forcejitterbuffer=yes
```

`iaxthreads` и `iaxmaxthreads` (глобальные)

Настройка `iaxthreads` определяет число вспомогательных потоков IAX, создаваемых при запуске для обработки соединений по протоколу IAX.

Настройка `iaxmaxthreads` определяет максимальное число дополнительных вспомогательных потоков IAX, которые могут быть созданы для обработки большого объема IAX-трафика:

```
iaxthreads=10
iaxmaxthreads=100
```

`jitterbuffer` (канал)

Неустойчивая синхронизация (Jitter) означает непостоянство задержки между поступлениями пакетов. Пакеты отправляются с конечного устройства с постоянной скоростью и практически постоянной задержкой между отправками. Однако при прохождении пакетов по Интернету задержка между ними может изменяться; из-за этого в место назначения пакеты поступают нерегулярно и, возможно, даже не в том порядке.

Буфер, компенсирующий задержки, в некотором смысле является промежуточной областью, в которой пакеты могут быть переупорядочены и направлены в место назначения регулярным потоком. Без такого буфера пользователь может почувствовать нарушения в потоке в виде помех, странных звуковых эффектов, искаженных слов или, в особо тяжелых случаях, пропущенных слов или слогов.

Буфер, компенсирующий задержки, обрабатывает только данные, поступающие с дальнего конца соединения. Данные, передаваемые вами, не будут затронуты, поскольку за компенсацию задержек в своих входящих соединениях отвечает дальний конец соединения.

Активация буфера, компенсирующего задержки, выполняется строкой `jitterbuffer=yes`:

```
jitterbuffer=yes|no
```

`language` **(канал)**

Задает флаг языка для всего, что вы определяете. Глобальный язык по умолчанию – английский. Заданный язык отправляется каналом как элемент информации. Он также используется такими приложениями, как `SayNumber()`, при выборе соответствующего файла для воспроизведения. Не забывайте, что все остальные языки, кроме английского, не устанавливаются в системе явно, поэтому ваша задача – сконфигурировать систему так, чтобы гарантировать правильную обработку задаваемых языков:

```
language=en
```

`mailboxdetail` **(глобальный)**

Если для `mailboxdetail` задано значение `yes`, пользователю вместо простого отчета о наличии новых и старых сообщений отправляется количество новых/старых сообщений. Параметр `mailboxdetail` также может быть задан для каждого равноправного участника сети отдельно:

```
mailboxdetail=yes
```

`maxjitterbuffer` **(канал)**

Этот параметр используется для задания максимального размера буфера, компенсирующего задержки, в миллисекундах. Не задавайте `maxjitterbuffer` слишком большое значение, это приведет к ненужному увеличению вашего времени ожидания:

```
maxjitterbuffer=500
```

`maxjitterinterps` **(канал)**

Максимальное количество пустых кадров вставки, которое должен возвратить подряд буфер, компенсирующий задержки. Поскольку некоторые клиенты не передают кадры CNG/DTX для обозначения паузы в разговоре, буфер, компенсирующий задержки, будет воспринимать такое количество пустых кадров как начало паузы. Это предотвращает появление искажений при длительной паузе:

```
maxjitterinterps=10
```

`maxregexpire` **и** `minregexpire` **(канал)**

Задают в секундах максимальный и минимальный временные интервалы для истечения срока действия регистрации:

```
maxregexpire=180
minregexpire=60
```

`mohinterpret` **(канал)**

Эта опция определяет, какой класс музыки во время ожидания должен воспроизводиться по этому каналу, если в диалплане для канала нет выражения `Set(CHANNEL(musicclass)=`*любой*`)`, определяющего

класс музыки, и одноранговый канал, удерживающий вызов, не предлагает класса музыки.

Если эта опция имеет значение `passthrough`, для оповещения вместо локального воспроизведения музыки во время ожидания всегда будет передаваться сообщение ожидания. Эта опция может быть задана глобально или для каждого пользователя либо равноправного участника сети в отдельности:

```
mohinterpret=default
```

`mohsuggest` **(канал)**

Эта опция определяет, какой класс музыки во время ожидания (как определено в файле musiconhold.conf) должен предлагаться одноранговому каналу, когда этот канал переводит равноправного участника сети в режим ожидания. Он может быть задан глобально или для каждого пользователя либо равноправного участника сети в отдельности:

```
mohsuggest=default
```

`nochecksums` **(глобальный)**

Если этот параметр задан, Asterisk деактивирует вычисление контрольных сумм UDP пакетов и не будет проверять контрольные суммы UDP от систем, поддерживающих эту функцию:

```
nochecksums=yes
```

`regcontext` **(канал)**

Задавая контекст, содержащий некоторые команды, можно сконфигурировать Asterisk на выполнение ряда действий при регистрации равноправного участника сети на вашем сервере. Эта опция используется в сочетании с `regexten`, определяющей, какой добавочный номер должен быть выполнен. Если параметр `regexten` не задан, в качестве добавочного номера используется имя равноправного участника. Asterisk будет динамически создавать и уничтожать для добавочного номера `NoOp` в приоритете 1. Все действия, которые будут выполняться при регистрации, должны начинаться с приоритета 2. Может быть задано несколько `regexten`, разделенных символом `&`. `regcontext` задается для каждого равноправного участника или глобально:

```
regcontext=зарегистрированные-телефоны
```

`regexten` **(канал)**

Опция `regexten` используется в сочетании с `regcontext` для определения добавочного номера, выполняемого в заданном контексте. Если опция `regexten` не задана явно, в качестве добавочного номера для сопоставления используется имя равноправного участника сети:

```
regexten=мойтелефон
```

`resyncthreshold` **(канал)**

Порог ресинхронизации используется для восстановления синхронизации буфера, компенсирующего задержки, если после получе-

ния нескольких кадров выявлено существенное отклонение, при условии что отклонение было обусловлено путаницей с временными метками. Порог ресинхронизации определяется как замеренная флуктуация[1] плюс значение `resyncthreshold` в миллисекундах:

```
resyncthreshold=1000
```

`rtautoclear` **(глобальный)**

Этот параметр определяет, должна ли Asterisk автоматически завершать действие регистрации соединений типа `friend`, созданных «на лету», по тому же графику, как если бы они были зарегистрированы обычным способом. Если задано значение `yes`, по истечении срока действия регистрации `friend` исчезнет из конфигурации до следующей регистрации. Если задано целое значение, регистрация будет действительна в течение этого количества секунд, а не обычного срока действия регистрации:

```
rtautoclear=yes|no|количествосекунд
```

`rtcachefriends` **(глобальный)**

Если `rtcachefriends` включен, Asterisk будет кэшировать соединения типа `friend`, регистрирующиеся в режиме реального времени, точно так же, как если бы они поступали из iax.conf. Это часто помогает в таких вопросах, как оповещение о непросмотренных сообщениях для равноправных участников сети, зарегистрировавшихся в режиме реального времени:

```
rtcachefriends=yes|no
```

`rtignoreregexpire` **(глобальный)**

Если для параметра `rtignoreregexpire` задано значение `yes` и срок регистрации равноправного участника сети, зарегистрировавшегося в режиме реального времени, истек (на основании срока действия регистрации), Asterisk продолжит использовать IP-адрес и порт, хранящиеся в базе данных:

```
rtignoreregexpire=yes|no
```

`rtupdate` **(глобальный)**

Если задано значение `yes`, Asterisk обновит при регистрации IP-адрес, порт вызова и срок действия регистрации равноправного участника сети. Значение по умолчанию – `yes`:

```
rtupdate=yes|no
```

`tos` **(глобальный)**

Asterisk может задавать в IP-заголовке биты типа обслуживания (Type of Service, TOS), чтобы способствовать повышению производительности в маршрутизаторах, которые учитывают TOS-биты при определении маршрутов. Допустимыми являются такие значения: `CS0`, `CS1`, `CS2`, `CS3`, `CS4`, `CS5`, `CS6`, `CS7`, `AF11`, `AF12`, `AF13`, `AF21`, `AF22`, `AF23`, `AF31`,

[1] Флуктуация – случайное отклонение величины, характеризующей систему из большого числа частиц, от ее среднего значения. – *Примеч. науч. ред.*

AF32, AF33, AF41, AF42, AF43 и ef (expedited forwarding – срочная пересылка). Также для TOS-битов может использоваться числовое значение.

Больше информации можно найти в файле doc/ip-tos.txt в папке исходного кода Asterisk.

`trunk` (канал)

Функция объединения каналов IAX2 позволяет Asterisk отправлять голосовые данные (в виде мини-кадров) из нескольких каналов под одним заголовком. Сокращение дополнительных издержек делает протокол IAX2 более эффективным при отправке нескольких потоков в одну конечную точку (обычно на другой сервер Asterisk):

```
trunk=yes|no
```

`trunkfreq` (канал)

`trunkfreq` используется для управления частотой отправки сообщений по магистральному каналу. Задается в миллисекундах. Сообщения отправляются вместе с командой `trunk=yes`:

```
trunkfreq=20
```

`trunktimestamps` (канал)

Определяет, должна ли Asterisk посылать временные метки для каждого отдельного подкадра, входящего в состав магистрального кадра (trunk frame). При передаче этих временных меток немного увеличивается требуемая полоса пропускания (менее чем на 1 Кбит/с

### Извлечение информации диалплана с удаленного сервера Asterisk

Asterisk может извлекать информацию диалплана с другого сервера Asterisk, используя выражение `switch =>`. Когда это происходит, драйвер IAX-канала Asterisk должен дождаться ответа от удаленного сервера, прежде чем сможет продолжать выполнение всех остальных связанных с IAX процессов. Это доставляет особенно много неприятностей, когда имеется множество вложенных выражений `switch`: если выражение `switch` указывает на несколько серверов, результат может возвратиться с ощутимой задержкой.

Если для глобальной опции `iaxcompat` задано значение `yes`, при осуществлении поиска `switch` Asterisk будет порождать отдельный поток. Использование этого потока позволяет драйверу IAX-канала продолжать выполнение других процессов, пока поток ожидает ответа. Эта опция обусловливает небольшие потери производительности:

```
iaxcompat=yes|no
```

на вызов), но они гарантируют надежную передачу временных меток кадров из конца в конец. Если оба конца всех ваших магистральных каналов направляются прямо в TDM и значение `trunkfreq` равно длине кадра используемого кодека, вероятно, можно опустить этот параметр. Получатель также должен поддерживать эту функцию, хотя необязательно, чтобы она была у него активирована:

```
trunktimestamps=yes|no
```

## Регистрация на других серверах с помощью выражений register

Выражение регистрации `register =>` используется для регистрации вашего сервера Asterisk на удаленном сервере. Это позволяет удаленному концу соединения знать ваше местонахождение на случай, если вы сконфигурированы с динамическим IP-адресом. Заметьте, что выражения `register` используются, только если вы сконфигурированы на удаленном конце как равноправный участник сети и когда `host=dynamic`.

Основной формат выражения `register`:

```
register => имяпользователя:пароль@удаленный-хост
```

*пароль* является необязательным параметром (если не сконфигурирован в удаленной системе).

В качестве альтернативы можно задать RSA-ключ, заключив его имя[1] в квадратные скобки ([ ]):

```
register => имяпользователя:[имя-rsa-ключа]@удаленный-хост
```

По умолчанию запросы `register` будут посылаться через порт 4569. Их можно направить на другой порт, явно задав его:

```
register => имяпользователя:пароль@удаленный-хост:1234
```

## Описания IAX-каналов

Теперь, когда определены общие настройки, можно приступать к описанию каналов. Если предполагается принимать анонимные IAX-вызовы, рекомендуется создать гостевой канал. Это очень распространенный способ связи в сообществе Asterisk. Прежде чем решить, что это не для вас, подумайте о том, что, если вы хотите предоставить кому-либо возможность связи с вами через IAX (без конфигурации специальной учетной записи), он должен соединяться как гость. Эта учетная запись, в сущности, становится вашим «телефонным номером по протоколу IAX». Описание гостевого канала должно выглядеть примерно так:

```
[guest]
type=user
context=incoming
callerid="Incoming IAX Guest"
```

[1] RSA-ключи Asterisk обычно располагаются в папке /var/lib/asterisk/keys/. С помощью сценария `astkeygen` можно генерировать собственные ключи.

Несомненно, спамеры найдут способ доступа к этим адресам, но в ближайшем обозримом будущем это не представляет проблемы. В долгосрочной перспективе, вероятно, будет использоваться DUNDi (больше информации об этом вы найдете в главе 14).

Если требуется принимать вызовы из сети Free World Dialup, в Asterisk есть предопределенный защитный ключ, который гарантирует невозможность для анонимных соединений сымитировать звонок Free World Dialup. Для этого потребуется настроить канал `iaxfwd`:

```
[iaxfwd]
type=user
context=incoming
auth=rsa
inkeys=freeworlddialup
```

Если имеются ресурсы, опубликованные в сети DUNDi, в файле iax.conf должен быть описан соответствующий пользователь:

```
[dundi]
type=user
dbsecret=dundi/secret
context=dundi-incoming
```

Если имеются устройства, работающие по протоколу IAX (такие, как IAXy), или IAX-пользователи на удаленном сервере, возможно, вы захотите обеспечить их собственным описанием пользователя, посредством которого они будут соединяться с системой.

Допустим, на удаленном сервере имеется пользователь, для которого решено определить IAX-канал типа `user`. Назовем этого гипотетического пользователя sushi. Описание этого канала может быть примерно таким:

```
[sushi]
type=user
context=local_users
auth=md5,plaintext,rsa
secret=wasabi
transfer=no
jitterbuffer=yes
callerid="Happy Tempura" <(800) 555-1234>
accountcode=seaweed
deny=0.0.0.0/0.0.0.0
permit=192.168.1.100/255.255.255.0
language=en
```

Входящие звонки от этого пользователя будут поступать в контекст `local_users` и передавать системе Caller ID (ID звонящего) `Happy Tempura <(800) 555-1234>`. Система ожидает от этого пользователя аутентификации с использованием простого текстового пароля или алгоритмов MD5 и RSA, поскольку предоставлен пароль wasabi и звонок поступил с IP-адреса 192.168.1.100. Всем звонкам, поступившим по этому каналу,

### Аутентификация по протоколу IAX

IAX предоставляет механизмы аутентификации для обеспечения достаточного уровня безопасности между конечными точками. Это не означает, что аудиоинформацию нельзя захватить и декодировать, но свидетельствует о том, что можно более тщательно управлять правами доступа к вашей системе. В IAX-каналах поддерживается три уровня безопасности. Опция `auth` определяет, какой метод аутентификации используется в канале: `plaintext`, `md5` или `rsa`.

Параметр `plaintext` в IAX обеспечивает очень низкий уровень защиты. Хотя он разрешает соединение с каналом только при условии введения правильного пароля, но тот факт, что пароль хранится в файле iax.conf как простой текст и передается и принимается в таком же незашифрованном виде, делает этот метод аутентификации очень ненадежным.

`md5` обеспечивает бóльшую безопасность сетевого соединения, однако по-прежнему в файле iax.conf на обоих концах соединения должен быть задан текстовый `secret`. Вот как происходит аутентификация в данном случае: сервер A запрашивает соединение с сервером B, который, в свою очередь, отвечает запросом на авторизацию, включающим сгенерированный случайным образом номер. Сервер A генерирует хеш MD5, используя значение, заданное в поле `secret` файла iax.conf, и случайный номер, полученный от сервера B. Этот хеш возвращается в ответе на запрос на авторизацию, и сервер B сравнивает его с локально сгенерированным хешем. Если хеши совпадают, предоставляется разрешение на доступ.

Метод `rsa` обеспечивает самый высокий уровень безопасности. Чтобы использовать RSA-аутентификацию, каждый конец соединения с помощью сценария astgenkey, обычно находящегося в папке /usr/src/asterisk/contrib/scripts/, должен создать пару ключей – открытый и закрытый. После этого открытый ключ передается на дальний конец. Каждый конец соединения в свое описание канала должен включить открытый ключ противоположного конца, используя для этого параметры `inkeys` и `outkey`.

RSA-ключи хранятся в папке /var/lib/asterisk/keys/. Открытым ключам присваиваются имена *имя*.pub; закрытым ключам – *имя*.key. Закрытые ключи должны быть зашифрованы по алгоритму 3DES[1].

[1] Из соображений безопасности файл закрытого ключа должен быть защищен от несанкционированного копирования. – *Примеч. науч. ред.*

будет присваиваться код учетной записи seaweed. Поскольку для параметра transfer (переадресация) задано значение no, медиа-поток этого канала всегда будет проходить через Asterisk; он не может быть перенаправлен на другой IAX-узел.

Если вы сами являетесь удаленным узлом и вам необходимо устанавливать соединения с другим узлом, главный узел был бы определен для вас как равноправный участник (peer):

```
[sashimi_platter]
type=peer
username=sushi
secret=wasabi
host=192.168.1.101
qualify=yes
trunk=yes
```

peer вызывается из диалплана с помощью приложения Dial(), в которое передается имя, указанное в квадратных скобках. Если равноправный участник требует от вас аутентификации с использованием имени пользователя, имя пользователя и секрет можно задать в полях username и secret.

Помните, аутентификация входящего звонка от пользователя, заданного в iax.conf, должна выполняться с использованием имени, заданного в квадратных скобках. Однако если Asterisk сама вызывает внешнего равноправного участника сети, имя пользователя, используемое при аутентификации, можно задать с помощью настройки username.

Для описания host используется или запись IP-адреса с точками-разделителями, или полное доменное имя (fully qualified domain name, FQDN). Задавая параметр qualify=yes, можно определить задержку соединения между вами и удаленным хостом и проверку, активен ли он. Чтобы свести до минимума количество издержек при поступлении множества звонков к одному равноправному участнику сети, их можно объединить (trunk).

*Объединение* каналов является уникальной возможностью IAX. Благодаря ей между двумя большими сайтами можно устанавливать множество одновременных VoIP-соединений. Объединение каналов, предоставляемое IAX, обеспечивает сокращение количества переносимой служебной информации за счет загрузки в каждый сигнальный пакет аудиоданных нескольких параллельных вызовов[1]. Чтобы активировать

[1] Можно провести аналогию между объединением VoIP-пакетов, реализуемым с помощью IAX, и объединением вагонов, принадлежащих различным компаниям, в один состав. Это очень полезно во многих ситуациях, поскольку объем служебных данных IP (UDP-заголовки, IP-заголовки и т. д.) часто превышает объем полезной нагрузки (аудиоинформации). Если между двумя серверами Asterisk выполняется несколько параллельных вызовов, вы обязательно захотите включить объединение каналов!

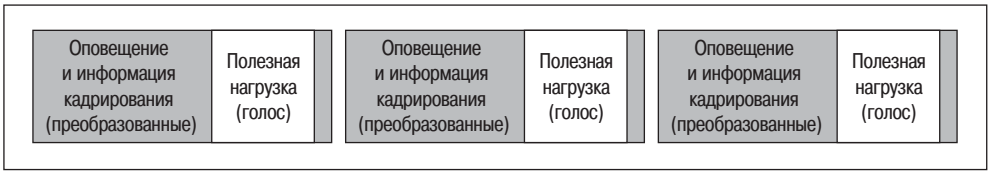



***Рис. А.1.** Объединение каналов деактивировано*


***Рис. А.2.** Объединение активировано*

объединение для канала, необходимо задать для него параметр `trunk=yes` в файле iax.conf.

На рис. А.1 показан канал с отключенной возможностью объединения, а на рис. А.2 – канал, для которого эта возможность активирована.

## Параметры канала

Теперь рассмотрим параметры канала:

`callerid`

С помощью параметра `callerid` можно задать рекомендуемый строковый Caller ID (ID звонящего) для каналов типа `user` или `peer`. Если для `user` задано значение в поле Caller ID, всем звонкам, поступающим по этому каналу, будет присвоен этот Caller ID, независимо от того, что посылает вам дальний конец соединения. Если Caller ID задан для `peer`, вы посылаете запрос дальнему концу на использование его как вашего идентификатора (хотя не располагаете средствами, чтобы проконтролировать это). Если вы хотите, чтобы вызывающие абоненты могли использовать собственные Caller ID (то есть для гостей), убедитесь, что значение в поле `callerid` не задано:

```
callerid=John Smith <(800) 555-1234>
```

`defaultip`

Настройка `defaultip` дополняет `host=dynamic`. Если хост еще не зарегистрирован на вашем сервере, вы будете пытаться отправлять сообщения по указанному здесь IP-адресу по умолчанию:

```
defaultip=192.168.1.101
```

`inkeys`

Опция `inkeys` может использоваться для аутентификации пользователя с помощью RSA-ключа. Чтобы связать с описанием канала типа `user` более одного RSA-ключа, имена ключей записываются через

двоеточие (:). Для подтверждения допустимости соединения может использоваться любой из заданных ключей. Inkey – это открытый ключ, который вы раздаете своим пользователям:

```
inkeys=сервер_один:сервер_два
```

`mailbox`

Если в описании канала вы связываете с `peer` почтовый ящик, сервис голосовой почты будет посылать сигналы индикации ожидающего сообщения (Message Waiting Indication, MWI) узлам на конце этого канала. Если номер почтового ящика обрабатывается в другом контексте голосовой почты, не `default`, его можно описать как *`почтовыйящик@контекст`*. Чтобы связать несколько почтовых ящиков с одним `peer`, используется несколько выражений `mailbox`:

```
mailbox=1000@internal
```

`outkey`

Опция `outkey` может использоваться для аутентификации канала `peer` с помощью RSA-ключа. Для исходящей аутентификации может использоваться только один RSA-ключ. Outkey не распространяется; это ваш закрытый ключ:

```
outkey=закрытый_ключ
```

`qualify`

Для параметра `qualify` может быть задано значение `yes`, `no` или время в миллисекундах. Если задается `qualify=yes`, удаленным равноправным участникам периодически будут посылаться сообщения `PING` для определения, доступны ли они, и установления величины задержки между ответами. Равноправные участники будут отвечать сообщениями `PONG`. Равноправный участник будет признан недоступным в случае непоступления ответа в течение 2000 мс (изменить это значение по умолчанию можно, задав для параметра `qualify` время ожидания ответа в миллисекундах).

`qualifyfreqok` и `qualifyfreqnotok`

Эти две настройки используются для определения частоты, с какой Asterisk будет отправлять сообщения `PING` равноправному участнику сети, если задан параметр `qualify`. Параметры `qualifyfreqok` и `qualifyfreqnotok` определяют частоту проверки возможности установления соединения с удаленным участником, когда он находится в состоянии OK и не в состоянии OK соответственно.

`qualifysmoothing`

Для параметра `qualifysmoothing` может быть задано значение `yes` или `no`. Если он активирован, Asterisk будет брать среднее из двух последних значений времени подтверждения возможности соединения. Это помогает не допустить появления равноправных участников, отмеченных как `LAGGED` (с запаздыванием), особенно в сети с потерями.

`sendani`

В сети PSTN с SS7 для идентификации вызывающего абонента используется автоматическое определение номера (Automatic Number Identification, ANI). Пользователю предоставляется Caller ID (ID звонящего). Caller ID формируется из ANI, поэтому их легко спутать. Блокировка Caller ID приводит к установлению флага конфиденциальности для ANI, но базовой сети все равно известен источник вызова:

```
sendani=yes
```

ANI используется уже некоторое время. Его первоначальным назначением было доставка на входящую станцию номера абонента, выполняющего междугородный звонок, для которого выставляется счет. В отличие от Caller ID, ANI не требует SS7, поскольку может передаваться с помощью DTMF. Также ANI не может быть заблокирован.

`transfer`

Параметру `transfer` может быть присвоено значение `yes`, `no` или `mediaonly`. Если задано `yes`, Asterisk, если может, будет выполнять переадресацию вызова с целью сократить путь пакета между двумя конечными точками. (Очевидно, что это не будет возможным, если Asterisk придется выполнять перекодировку или преобразование между протоколами или если состояние сети не обеспечивает возможности соединения двух конечных точек напрямую.) Если задано значение `no`, Asterisk не будет пытаться переадресовать вызов.

Если задано значение `mediaonly`, Asterisk будет пытаться переадресовывать медиа-поток так, чтобы он проходил непосредственно между двумя конечными точками, но обмен служебными сигналами (сообщения установления и разрыва соединения) по-прежнему будет проходить через Asterisk. Это полезно, потому что гарантирует правильность записей параметров вызовов даже несмотря на то, что переносимые данные больше не проходят через сервер Asterisk.

## SIP

Как и для IAX, конфигурационный файл SIP (sip.conf) содержит информацию о конфигурации для каналов, работающих по протоколу SIP. Заголовки описаний каналов формируются словом, заключенным в квадратные скобки (`[ ]`), опять же, за исключением раздела `[general]`, в котором задаются глобальные параметры SIP. Не скупитесь на комментарии в файле sip.conf. Текст комментария начинается с точки с запятой; все, что располагается справа от нее, будет проигнорировано.

## Общие параметры SIP

В разделе [general] файла sip.conf должны использоваться следующие опции:

allowexternalinvites

Если задано значение no, эта настройка деактивирует отправку сообщений INVITE и REFER нелокальным доменам. Смотрите настройку domain.

```
allowexternalinvites=yes|no
```

allowguest

Если задано значение no, этот параметр запрещает гостевые SIP-соединения. По умолчанию они разрешены. SIP обычно требует аутентификацию, но можно принимать вызовы от пользователей, которые не поддерживают аутентификацию (то есть для которых не задано значение в поле secret). Некоторые SIP-устройства (такие, как Cisco Call Manager v4.1) не поддерживают аутентификацию, поэтому они не смогут устанавливать соединения, если задано allowguest=no:

```
allowguest=no|yes
```

allowoverlap

Если задано значение no, деактивирован набор в режиме наложения:

```
allowoverlap=no|yes
```

allowsubscribe

Разрешает или запрещает внешним устройствам подписываться на получение информации о состоянии добавочного номера (заданное в приоритете hint). Значение по умолчанию – yes:

```
allowsubscribe=yes|no
```

allowtransfers

Значение no деактивирует переадресацию для всех SIP-вызовов, кроме тех, для которых она активирована специально:

```
allowtransfers=no|yes
```

alwaysauthreject

Если эта опция активирована, любое отклонение INVITE или REGISTER Asterisk будет сопровождать сообщением 401 Unauthorized, а не предоставлять вызывающему абоненту информацию о наличии соответствующего user или peer для этого запроса:

```
alwaysauthreject=no|yes
```

autodomain

Задайте для этой опции значение yes, тогда Asterisk будет добавлять имя локального хоста и локальные IP-адреса в список доменов:

```
autodomain=yes|no
```

`bindaddr` и `bindport`

Эти необязательные параметры позволяют задавать IP-интерфейс и порт, на которые вы желаете принимать SIP-соединения. Если данные параметры опущены, будет задан порт 5060 и все IP-адреса вашей системы Asterisk будут принимать входящие SIP-соединения. Если задано несколько адресов привязки, соединения будут слушать только эти интерфейсы. Значение `0.0.0.0` указывает Asterisk слушать все интерфейсы:

```
bindaddr=0.0.0.0
bindport=5060
```

`buggymwi`

Эта настройка позволяет Asterisk отправлять уведомления об ожидающих сообщениях на определенные SIP-телефоны Cisco, прошивка которых не обеспечивает полной поддержки индикации ожидающих сообщений, описанной в документации RFC для Интернета. Активируйте эту опцию, чтобы не получать сообщения об ошибках при отправке MWI-сообщений на телефоны с таким недостатком:

```
buggymwi=no|yes
```

`callevents`

Задайте значение `yes`, если вы хотите, чтобы SIP формировал события интерфейса Manager. Это важно при наличии внешних программ, использующих интерфейс Asterisk Manager, таких как Flash Operator Panel:

```
callevents=yes
```

`checkmwi`

Эта опция определяет интервал времени по умолчанию, в секундах, между проверками почтовых ящиков для равноправных участников:

```
checkmwi=30
```

`compactheaders`

Для параметра `compactheaders` можно задать значение `yes` или `no`. Если задано `yes`, для SIP-заголовков будет использоваться компактный формат. Это может потребоваться, когда размер SIP-заголовка больше максимального размера передаваемого блока данных (Maximum Transmission Unit, MTU) ваших IP-заголовков, что приводит к фрагментации IP-пакета. Не используйте эту опцию, если не знаете, что делаете:

```
compactheaders=yes|no
```

`defaultexpiry`

Задает срок действия SIP-регистрации по умолчанию, в секундах, для входящих и исходящих регистраций. Клиент обычно задает это значение при первой регистрации, поэтому значение по умолчанию

будет использоваться, только если клиент не задал собственное значение. Если вы регистрируетесь на другом сервере агента пользователя (User Agent Server, UAS), этот срок регистрации будет передан на дальний конец:

```
defaultexpiry=300
```

`directrtpsetup`

Эта настройка конфигурирует прямое установление соединения в реальном масштабе времени между двумя конечными точками без необходимости повторного обмена сообщениями `INVITE`.

```
directrtpsetup=yes|no
```

На момент написания данной книги параметр `directrtpsetup` все еще считается экспериментальным, и поэтому вы не должны активировать его, если полностью не осознаете последствий этого действия. Эта опция не будет работать для видеосессий и в случаях, когда вызываемая сторона посылает полезную нагрузку RTP и FMTP-заголовки в ответе `200 OK`, что не соответствует запросу `INVITE` вызывающего абонента.

`domain`

Определяет домен по умолчанию для данного сервера Asterisk. Если определен этот параметр, Asterisk допускает отправку сообщений `INVITE` и `REFER` только нелокальным доменам. Получить список локальных доменов можно с помощью CLI-команды `sip show domains`:

```
domain=example.com
```

`dumphistory`

Для параметра `dumphistory` можно задать значение `yes` или `no`, чтобы активировать или деактивировать вывод отчета по истории SIP в конце диалогового окна SIP. SIP-история записывается в канал протоколирования `DEBUG`:

```
dumphistory=yes|no
```

`externhost`

Параметр `externhost` принимает в качестве аргумента полное имя домена. Если Asterisk выполняется за NAT, SIP-заголовок, как правило, будет использовать внутренний IP-адрес, присвоенный серверу. Если вы зададите эту опцию, Asterisk будет периодически выполнять DNS-поиск по имени хоста и замещать внутренний IP-адрес на тот, который был возвращен в результате DNS-поиска:

```
externhost=my.hostname.tld
```

В системах, находящихся в производственной эксплуатации, не рекомендуется использовать `externhost`, потому что в случае изменения IP-адреса сервера в SIP-заголовках будет указываться неверный IP-адрес вплоть до следующего поиска. Вместо этого рекомендуется использовать параметр `externip`.

`externip`

Параметр `externip` в качестве аргумента принимает IP-адрес. Если Asterisk выполняется за NAT, SIP-заголовок будет использовать внутренний IP-адрес, заданный для сервера. Удаленный сервер не будет знать, как вернуться к этому адресу; поэтому он должен быть заменен действительным маршрутизируемым адресом:

```
externip=216.239.39.104
```

`externrefresh`

Если используется `externhost`, `externrefresh` определяет, сколько времени, в секундах, должно пройти между DNS-поисками:

```
externrefresh=30
```

`g726nonstandard`

Этот параметр может быть задан при общении с равноправными участниками, которые ошибочно используют неверную кодировку для кодека G.726. Эта настройка указывает Asterisk использовать порядок упаковки по протоколу AAL2, а не RFC3551, если равноправный участник согласовывает использование кодека G726-32. Обычно это противоречит спецификации RFC3551, поскольку равноправный участник должен согласовывать использование AAL2-G726-32. Эта опция может понадобиться в случае применения устройства Sipura или Grandstream:

```
g726nonstandard=yes
```

`ignoreregexpire` (глобальный)

Если параметр `ignoreregexpire` имеет значение `yes`, Asterisk может выполнить одно из двух действий для:

*Равноправных участников, создаваемых не в режиме реального времени*

По истечении срока их регистрации информация не будет удалена из памяти или базы данных Asterisk. При попытке вызова данного равноправного участника существующая информация будет использоваться, несмотря на истечение ее срока действия.

*Равноправных участников, создаваемых в режиме реального времени*

Когда равноправный участник извлекается из хранилища реального времени, его регистрационная информация будет использоваться независимо от истечения ее срока действия; если срок ее действия истек, в то время как равноправный участник, созданный в режиме реального времени, все еще находится в памяти (из-за кэширования или по другим причинам), информация не будет удалена из хранилища реального времени:

```
ignoreregexpire=yes|no
```

`jbenable`

Активирует использование RTP-буфера, компенсирующего задержки, на принимающей стороне SIP-канала. Значение по умолчанию – `no`. Активированный буфер, компенсирующий задержки, будет использоваться, только если отправляющая сторона может создавать неустойчивую синхронизацию, а принимающая сторона ее не допускает. SIP-канал допускает неустойчивую синхронизацию; таким образом, компенсирующий задержки буфер на принимающей стороне будет использоваться, только если он активирован и задано его принудительное использование:

```
jbenable=yes|no
```

`jbforce`

Обусловливает принудительное использование RTP-буфера, компенсирующего задержки, на принимающей стороне SIP-канала. Значение по умолчанию – `no`:

```
jbforce=yes|no
```

`jbimpl`

Эта настройка используется для задания типа используемого буфера, компенсирующего задержки: `fixed` (фиксированный) или `adaptive` (адаптивный). Если используется буфер `fixed`, его размер всегда будет равен тому, который определен параметром `jbmaxsize`. Если задан буфер `adaptive`, его размер будет меняться вплоть до максимального, определенного параметром `jbmax size`. Значение по умолчанию – `fixed`.

```
jbimpl=fixed|adaptive
```

`jblog`

Определяет, активирована или нет запись в журнал кадров буфера, компенсирующего задержки. Значение по умолчанию – `no`:

```
jblog=yes|no
```

`jbmaxsize`

Задает максимальный размер буфера, компенсирующего задержки, в миллисекундах:

```
jbmaxsize=200
```

`jbresyncthreshold`

Переходит на временные метки кадров, из-за которых произошла рассинхронизация буфера, компенсирующего задержки. Полезно для улучшения качества голоса, переданного со скачкообразными/прерывистыми временными метками, которые обычно поступают с экзотических устройств и программ. Значение по умолчанию – `1000`:

```
jbresyncthreshold=1000
```

limitonpeers

Эта настройка указывает Asterisk применять ограничения по количеству вызовов только к равноправным участникам. Это улучшит уведомление о допустимом количестве вызовов и статусе для устройств типа type=friend, потому что будет контролироваться предельное число вызовов peer и не будут создаваться отдельные счетчики для частей user и peer канала friend:

```
limitonpeers=yes|no
```

localnet

Параметр localnet используется для указания Asterisk, какие IP-адреса считать локальными, чтобы адрес в SIP-заголовке мог транслироваться в заданный в externip или чтобы можно было выполнять поиск IP-адреса по externhost. IP-адреса должны задаваться в нотации CIDR (Classless InterDomain Routing – бесклассовая междоменная маршрутизация):

```
localnet=192.168.1.0/24
localnet=172.16.0.0/16
```

matchexterniplocally

Определяет, что Asterisk должна подставлять настройку externip или externhost, только если она совпадает с вашей настройкой localnet. Активировать эту опцию потребуется только для сетей с очень необычными настройками:

```
matchexterniplocally=yes|no
```

maxexpiry

Задает максимальный срок, в секундах, действия регистрации равноправного участника:

```
maxexpiry=3600
```

minexpiry

Задает минимально допустимую продолжительность, в секундах, регистрации или подписки:

```
minexpiry=60
```

notifymimetype

Принимает в качестве аргумента строку, определяющую тип MIME (Multipurpose Internet Mail Extensions – многоцелевые почтовые расширения в Интернете), используемый для индикации ожидающего сообщения в SIP-сообщении NOTIFY. Чаще всего для этого поля применяется настройка text/plain, хотя в случае необходимости может использоваться и другое значение:

```
notifymimetype=text/plain
```

notifyringing

Определяет, должна ли Asterisk уведомлять подписчиков о состоянии RINGING:

```
notifyringing=yes|no
```

`notifyhold`

Определяет, должна ли Asterisk уведомлять подписчиков о состоянии `HOLD`:

```
notifyhold=yes|no
```

`pedantic`

Для параметра `pedantic` может быть задано значение `yes` или `no`. Значение `yes` активирует медленную, педантичную проверку для телефонов, которым она необходима, таких как Pingtel, и более строгое соответствие SIP RFC. С целью повышения производительности строгий контроль соответствия SIP RFC обычно не проводится:

```
pedantic=yes
```

`realm`

Эта опция задает область действия краткой аутентификации. Задайте в качестве значения `realm` свое полное доменное имя, которое должно быть глобально уникальным:

```
realm=myserver.example.com
```

`recordhistory`

Можно задать для `recordhistory` значение `yes` или `no`, чтобы активировать или отключить запись SIP-истории для всех каналов:

```
recordhistory=yes|no
```

`registerattempts`

Определяет для Asterisk количество попыток исходящих регистраций. Значение по умолчанию – 0, что означает бесконечное число попыток.

```
registerattempts=0
```

`registertimeout`

Определяет, как часто Asterisk должна выполнять попытку повторно зарегистрироваться на других устройствах:

```
registertimeout=30
```

`relaxdtmf`

Для параметра `relaxdtmf` можно задать значение `yes` или `no`. Значение `yes` обусловит ослабление выявления DTMF-сигналов. Оно должно использоваться, если Asterisk испытывает трудности по определению наличия DTMF в SIP-канале. Обратите внимание, что это может приводить к «ложным срабатываниям», когда Asterisk ошибочно определяет наличие DTMF-сигнала при его отсутствии:

```
relaxdtmf=yes|no
```

`rtautoclear` (глобальный)

Определяет, должна ли Asterisk автоматически завершать действие регистрации соединений типа `friend`, созданных «на лету», по тому же графику, как если бы они зарегистрировались в обычном режи-

ме. Если задано значение yes, по истечении срока действия регистрации friend исчезнет из конфигурации до следующей регистрации. Если задано целое значение, регистрация будет действительна в течение этого количества секунд, а не в течение обычного срока действия регистрации:

```
rtautoclear=yes|no|количествосекунд
```

rtcachefriends (глобальный)

Если rtcachefriends включен, Asterisk будет кэшировать соединения типа friend, регистрирующиеся в режиме реального времени, точно так же, как если бы они поступали из iax.conf. Это часто помогает в таких вопросах, как оповещение о непросмотренных сообщениях для равноправных участников сети, зарегистрировавшихся в режиме реального времени:

```
rtcachefriends=yes|no
```

rtsavesysname (глобальный)

Определяет, должна ли Asterisk сохранять имя системы в базе данных реального времени в момент регистрации:

```
rtsavesysname=yes|no
```

rtupdate (глобальный)

Если задано значение yes, Asterisk будет обновлять IP-адрес, порт вызова и срок регистрации при регистрации равноправного участника сети. Значение по умолчанию – yes:

```
rtupdate=yes|no
```

sipdebug

Определяет, должна ли включаться отладка SIP с того момента, когда Asterisk загружает драйвер SIP-канала:

```
sipdebug=yes|no
```

sendrpid

Определяет, должна ли Asterisk посылать заголовок Remote-Party-ID (идентификатор удаленной стороны):

```
sendrpid=yes|no
```

srvlookup

SRV-записи DNS – это средство задания логических разрешимых адресов, по которым с вами можно связаться. Позволяет перенаправлять вызовы в разные точки без необходимости изменения логического адреса. Использование SRV-записей открывает доступ ко многим преимуществам DNS, тогда как их отключение лишает вас возможности размещать SIP-вызовы на основании доменных имен.

В настоящее время поддержка SRV-записей в Asterisk несколько неэффективна. Если возвращено несколько SRV-записей, Asterisk будет использовать только первую из них.

Настоятельно рекомендуется использование DNS-поиска SRV-записей. Чтобы активировать его, задайте `srvlookup=yes` в разделе `[general]` файла sip.conf:

```
srvlookup=yes
```

`t1min`

Это минимальное время на передачу и подтверждение приема для сообщений, отправленных к контролируемым хостам, в миллисекундах. Значение по умолчанию – 100 мс.

```
t1min=100
```

`subscribecontext`

Ограничивает количество запросов `SUBSCRIBE` (подписаться) к заданному контексту. Полезно, например, если необходимо ограничить количество подписок на внутренние добавочные номера. Эта опция также может быть задана для каждого пользователя или равноправного участника сети отдельно:

```
subscribecontext=internal
```

`t38pt_udptl`

Если для `t38pt_udptl` задано значение `yes`, активирована возможность транзитной пересылки факсов по протоколу T.38 (UDPTL) в вызовах от SIP к SIP при условии, что обе стороны поддерживают T.38. Чтобы передача факсов была возможна, эта настройка должна быть активирована в разделе `[general]` для всех устройств. Затем ее можно деактивировать для каких-то отдельных устройств:

```
t38pt_udptl=yes|no
```

Транзитная пересылка факсов по протоколу T.38 возможна только при вызовах от SIP к SIP, при этом не используются локальные каналы или каналы агента. Asterisk в настоящее время не может начинать или завершать передачу факсов по протоколу T.38; она может только выполнять транзитную пересылку UDPTL с одного устройства на другое.

`tos_sip`, `tos_audio` и `tos_video`

Asterisk может задавать биты TOS в IP-заголовке, чтобы улучшить производительность маршрутизаторов, которые учитывают биты TOS при определении маршрутов. Настройки `tos_sip`, `tos_audio` и `tos_video` управляют TOS-битами для SIP-сообщений, аудио- и видеоданными RTP соответственно. Допустимые значения: `CS0`, `CS1`, `CS2`, `CS3`, `CS4`, `CS5`, `CS6`, `CS7`, `AF11`, `AF12`, `AF13`, `AF21`, `AF22`, `AF23`, `AF31`, `AF32`, `AF33`, `AF41`, `AF42`, `AF43` и `ef` (срочная пересылка). Также в качестве TOS-битов можно использовать числовое значение.

Больше информации можно найти в файле doc/ip-tos.txt в папке исходного кода Asterisk.

`trustrpid`

Определяет, должна ли Asterisk доверять значению в заголовке `Remote-Party-ID`:

```
trustrpid=yes|no
```

`useragent`

Параметр `useragent` принимает в качестве аргумента строку, определяющую значение поля `useragent` в SIP-заголовке. Значение по умолчанию – `asterisk`:

```
useragent=Asterisk PBX v1.4
```

`usereqphone`

Опция `usereqphone` указывает Asterisk добавлять `;user=phone` в SIP URI, содержащие действительный номер телефона:

```
usereqphone
```

`videosupport` (оба)

Параметру `videosupport` можно задать значение `yes` или `no`. Если активирована общая поддержка видео, ее можно отключить для отдельного равноправного участника сети, но ее нельзя активировать для одного равноправного участника сети, если она не активирована в разделе `[general]`:

```
videosupport=yes|no
```

`vmexten`

Эта опция задает добавочный номер диалплана для доступа к ящику голосовой почты, который будет передан в разделе `Message-Account` сообщения `MWI NOTIFY`. Задавайте эту опцию, если ваше SIP-устройство поддерживает настройку `Message-Account`. Значение по умолчанию – `asterisk`:

```
vmexten=8500
```

## Настройки SIP-канала

После рассмотрения глобальных параметров SIP обсудим параметры канала. Они могут быть определены для пользователя, равноправного участника сети или для обоих (что указывается в скобках):

`accountcode` (для обоих)

Код учетной записи может определяться для каждого пользователя. Если задан, этот код учетной записи будет присваиваться записи вызова, когда не задан код учетной записи конкретного пользователя. Заданное имя `accountcode` будет использоваться как имя файла в формате CSV в папке /var/log/asterisk/cdr-csv/, где хранятся CDR для пользователей/равноправных участников сети/друзей:

```
accountcode=iax-имяпользователя
```

`allow` и `disallow` (для обоих)

Могут быть разрешены или запрещены определенные кодеки, что позволяет разработчику системы задавать перечень используемых кодеков. `allow` и `disallow` также могут быть определены для канала отдельно. Помните, что выражения `allow` в разделе `[general]` будут распространяться на все каналы, для которых не переопределено `disallow=all`. Согласование кодеков ведется в порядке их задания. Лучшей практикой считается определять `disallow=all`, а затем с помощью выражений `allow` явно задавать каждый кодек, который вы желаете использовать. Если ничего не задано, предполагается, что `allow=all`:

```
disallow=all
allow=ulaw
allow=gsm
allow=ilbc
```

`amaflags` (для обоих)

Система автоматической регистрации сообщений (Automatic Message Accounting, AMA) описана в документации компании Telcordia, в разделе FR-AMA-1. Эти документы определяют стандартные механизмы формирования и передачи CDR. Можно задать один из четырех флагов AMA (`default`, `omit`, `billing` или `documentation`), который будет применяться ко всем SIP-соединениям:

```
amaflags=documentation
```

`callerid` (для обоих)

С помощью параметра `callerid` можно задать рекомендуемый строковый Caller ID (ID звонящего) для каналов типа `user` или `peer`. Если для `user` задано поле Caller ID, всем звонкам, поступающим по этому каналу, будет присвоен этот Caller ID независимо от того, что посылает вам дальний конец соединения. Если оно задано для `peer`, вы посылаете запрос дальнему концу на использование этого Caller ID как вашего идентификатора (хотя не располагаете средствами, чтобы проконтролировать это). Если вы хотите, чтобы вызывающие абоненты могли использовать собственные Caller ID (то есть для гостей), убедитесь, что поле `callerid` не задано:

```
callerid=John Smith <(800) 555-1234>
```

`callgroup` и `pickupgroup` (для обоих)

Параметр `callgroup` используется для назначения описания канала одной или более группам. Опция `pickupgroup` может использоваться в сочетании с этим параметром, чтобы обеспечить возможность ответа на звонок на данный телефон с другого добавочного номера. Опция `pickupgroup` используется для определения, вызовы каких групп вызовов может принимать канал, – каналу предоставляется возможность отвечать на вызовы другого канала, если он входит в ту же группу `pickupgroup`, что и группа вызовов вызываемого канала. По умолчанию перехватить вызовы удаленных добавочных номеров можно, набрав *8 (это можно настроить в файле features.conf):

```
callgroup=1,3-5
pickupgroup=1,3-5
```

`callingpres` **(для обоих)**

Задает публикацию Caller ID для данного пользователя/равноправного участника сети. Эта настройка принимает одну из следующих опций:

`allowed_not_screened`

Публикация разрешена, экранирование учетных данных не производится.

`allowed_passed_screen`

Публикация разрешена, экранирование разрешено.

`allowed_failed_screen`

Публикация разрешена, экранирование запрещено.

`allowed`

Публикация разрешена, сетевой номер.

`prohib_not_screened`

Публикация запрещена, экранирование учетных данных не производится.

`prohib_passed_screen`

Публикация запрещена, экранирование разрешено.

`prohib_failed_screen`

Публикация запрещена, экранирование запрещено.

`prohib`

Публикация запрещена, сетевой номер.

`unavailable`

Номер недоступен.

```
=yes|no
```

`canreinvite` **(для обоих)**

SIP-протокол пытается соединить конечные точки напрямую. Однако Asterisk должна оставаться на линии передачи между конечными точками, если необходимо определять наличие DTMF (более подробную информацию можно найти в главе 4):

```
canreinvite=no
```

`context` **(для обоих)**

Контекст задается в описании канала, чтобы входящие звонки направлялись в соответствующий контекст в extensions.conf, где осуществляется их обработка (см. главы 4 и 5). Для любого канала, соединяющегося с сервером Asterisk, должен быть задан контекст,

в который он будет направлен. Контекст обязателен для любого описания канала типа `user`; если контекст не задан, входящие звонки будут направляться в контекст `default`:

```
context=incoming
```

Необходимо знать о необычном сценарии, который потребует задания контекста для `peer`. Когда по SIP-каналу поступает вызов, он сначала пытается найти соответствующее описание `user` (согласно имени пользователя, заданному в квадратных скобках, и паролю). Если он не может найти ни одного подходящего пользователя, он ищет соответствие среди `peer` по IP-адресу, с которого поступил вызов. Поскольку обычно `peer` не имеют контекстов, такой вызов в итоге поступит в контекст `default`. Хотя это будет работать, контекст `default` не следует использовать для обработки входящих звонков. Выход – определить контекст для каждого `peer`, который может обрабатывать входящие звонки. Чтобы поэкспериментировать с этим, можете позвонить на свой номер в службе Free World Dialup; вызов вернется прямо к вам.

`defaultip` (равноправный участник)

Настройка `defaultip` дополняет `host=dynamic`. Если хост еще не зарегистрирован на вашем сервере, вы будете пытаться отправлять сообщения по указанному здесь IP-адресу по умолчанию:

```
defaultip=192.168.1.101
```

`deny` (для обоих)

С помощью опции `deny` можно задавать конкретные IP-адреса и диапазоны. Чтобы ограничить доступ для диапазона IP-адресов, используется маска подсети, например `deny=192.168.1.0/255.255.255.0`. Также можно запретить все адреса, задав `deny=0.0.0.0/0.0.0.0`, а затем с помощью команды `permit` разрешить доступ только определенным адресам. Помните о влиянии на безопасность, которое оказывает данная настройка (см. также `permit`):

```
deny=0.0.0.0/0.0.0.0
```

`disallow` (для обоих)

См. `allow`.

`dtmfmode` (для обоих)

Параметру `dtmfmode` могут быть присвоены значения `inband`, `rfc2833` или `info`. DTMF-коды могут быть отправлены или в полосе частот (как часть аудиопотока), или вне полосы (как сигнальная информация) с помощью методов RFC 2833 или INFO. Метод `inband` работает надежно только при использовании кодека без сжатия, такого как G.711, μlaw или alaw. Рекомендуемым является метод `rfc2833`; однако некоторые устройства, например производимые компанией Grandstream, поддерживают метод `info`:

```
dtmfmode=rfc2833
```

В Asterisk 1.4 был введен DTMF-сигнал переменной длительности (Variable Length DTMF), чтобы Asterisk имела возможность сообщать дальнему концу соединения верную длительность нажатия кнопки на телефоне, подключенном к входящему каналу (согласно RFC 2833 IETF). Более старые системы Asterisk не понимают параметра переменной длины. В старых системах Asterisk DTMF-сигнал, доставляемый через RFC 2833, может быть неправильно интерпретирован, что приводит к странным эффектам, например, в сеансах передачи голосовой почты. Если требуется, чтобы настройка `rfc2833` была реализована, как в более старой версии (до 1.4), необходимо добавить опцию `rfc2833compensate=yes` в описание `peer` в файле sip.conf, который определяет порядок обмена информацией с вашей системой Asterisk версии до 1.4.

`fromdomain` (равноправный участник)

Позволяет задавать домен в поле `From:` SIP-заголовка. Может требоваться некоторыми поставщиками сервисов для аутентификации:

```
fromdomain=my.hostname.tld
```

`fromuser` (равноправный участник)

Позволяет задавать имя пользователя для аутентификации. Обычно используется имя, заключенное в квадратные скобки в описании канала, но оно может быть переопределено с помощью опции `fromuser`. Это позволяет обращаться к описанию канала по имени, отличному от того, которое используется для аутентификации:

```
fromuser=john_smith
```

`host` (равноправный участник)

Конфигурирует хост, с которым должен соединяться данный равноправный участник сети. Используйте полное доменное имя:

```
host=remote.hostname.tld
```

`incominglimit` (для обоих)

Эта опция ограничивает общее число одновременных звонков для равноправного участника сети или пользователя. Задает максимальное число одновременных исходящих звонков для равноправного участника сети или максимальное число входящих звонков для пользователя.

```
incominglimit=3
```

`insecure` (для обоих)

При получении сообщения `INVITE` от удаленного ресурса Asterisk пытается аутентифицировать строку символов перед знаком `@` в строке `INVITE`, полученную в SIP-заголовке с именем описания канала из sip.conf. Если удаленный конец связи является агентом пользователя, его аутентификация будет проводиться исходя из описания `user`. Однако, если удаленный конец является прокси-сервисом SIP, он будет аутентифицироваться по записи `peer`. Когда вызовы поступают от такого провайдера, как Free World Dialup, который выступает

в роли прокси для удаленной стороны, фактически вызывающей вас, этот провайдер не может принимать вызов от лица конечной точки. Поскольку было бы непрактичным конфигурировать аутентификацию для каждого FWD-пользователя и поскольку FWD не может отвечать на сообщения 407 Proxy Authentication Required (Необходима аутентификация на прокси), требуется альтернативный способ разрешения приема звонков от этих абонентов.

Задавая `insecure=invite`, вы определите, какому каналу `peer` ищется соответствие при сравнении IP-адреса или имени хоста и номера порта с предоставленными в поле `Contact` SIP-заголовка опциями `host` и `port` в sip.conf. Если соответствие найдено, исходное сообщение `INVITE` не станет требовать аутентификации и звонок будет разрешен.

При наличии большого количества конечных точек за NAT-устройством необходимо активировать параметр `insecure=port`, чтобы выполнять сопоставление только по IP-адресу. Чтобы не предъявлять требование на аутентификацию во входящем `INVITE` для `peer`, задайте `insecure=invite,port`:

```
insecure=invite
```

`language` (для обоих)

Задает флаг языка для всего, что вы определяете. Глобальный язык по умолчанию – английский. Заданный язык отправляется каналом как элемент информации. Он также используется такими приложениями, как `SayNumber()`, чтобы выбрать соответствующий файл для воспроизведения. Не забывайте, что все остальные языки, кроме английского, не устанавливаются в системе явно, поэтому ваша задача конфигурировать систему так, чтобы гарантировать правильную обработку задаваемых языков:

```
language=en
```

`mailbox` (равноправный участник)

Если в описании канала вы связываете `mailbox` с `peer`, сервис голосовой почты будет посылать MWI-сигналы узлам на конце этого канала. Если номер почтового ящика обрабатывается в другом контексте голосовой почты, не `default`, его можно описать как *`почтовыйящик`*`@`*`контекст`*. Чтобы связать несколько почтовых ящиков с одним `peer`, используется несколько выражений `mailbox`:

```
mailbox=1000@internal
```

`maxcallbitrate` (для обоих)

Задает максимальную скорость передачи данных для отдельного звонка от конкретного пользователя или к конкретному равноправному участнику сети. Значение по умолчанию – 384 Кбит/с:

```
maxcallbitrate=384
```

`md5secret` (для обоих)

Если вы не хотите использовать простые текстовые пароли в файлах sip.conf, с помощью `md5secret` можно сконфигурировать хеш MD5,

который будет использоваться для аутентификации. Чтобы сгенерировать хеш MD5 из консоли Linux, используйте следующую команду:

```
# echo -n "username:realm:secret" | md5sum
```

Не забудьте использовать флаг -n, иначе echo добавит \n в конец строки; тогда символ перевода строки будет учтен при вычислении хеша MD5, что приведет к созданию неверного хеша. Если не задана опция realm (обсуждаемая в списке общих параметров SIP), будет принята область действия по умолчанию – asterisk. Если в одном описании канала заданы параметры и md5secret, и secret, последний будет проигнорирован:

```
md5secret=0bcbe762982374c276fb01af6d272dca
```

`mohinterpret` **(канал)**

Эта опция определяет, какой класс музыки во время ожидания должен воспроизводиться по данному каналу, если в диалплане для канала нет выражения Set(CHANNEL(musicclass)=*любой*), определяющего класс музыки, и канал типа peer, поддерживающий вызов, не предлагает класса музыки.

Эта опция может быть задана глобально или для каждого пользователя либо равноправного участника сети в отдельности:

```
mohinterpret=default
```

`mohsuggest` **(канал)**

Эта опция определяет, какой класс музыки во время ожидания (как определено в musiconhold.conf) должен предлагаться каналу типа peer, когда этот канал переводит равноправного участника сети в режим ожидания. Он может быть задан глобально или для каждого пользователя или равноправного участника сети в отдельности:

```
mohsuggest=default
```

`musicclass` **(для обоих)**

Эта опция задает класс музыки во время ожидания по умолчанию:

```
musicclass=classical
```

`nat` **(для обоих)**

Для параметра nat может быть задано значение yes, no или never. Если задано yes, Asterisk игнорирует IP-адрес в заголовках SIP и SDP и отвечает на адрес и порт, указанные в IP-заголовке. Опция never предназначена для устройств, которые не могут обрабатывать поле rport в SIP-заголовке, такое как Uniden UIP200:

```
nat=yes|no|never
```

`permit` **(для обоих)**

См. deny.

`pickupgroup` **(для обоих)**

См. callgroup.

`port` (равноправный участник)

Этот параметр может использоваться для задания порта, по которому будут слушаться SIP-сигналы, если вы хотите использовать для этого нестандартный порт. (Порт по умолчанию для обмена сигналами по протоколу SIP – 5060.)

```
port=5060
```

`progressinband` (для обоих)

Для параметра `progressinband` может быть задано значение `yes`, `no` или `never`, чтобы определить, должна ли Asterisk самостоятельно генерировать звуковой сигнал вызова для вызываемого абонента. Обычно Asterisk использует для информирования о поступлении вызова несколько методов, таких как 183 Session Progress, 180 Ringing, 486 Busy и т. д. Если задано `progressinband=yes`, Asterisk будет генерировать тональные сигналы для обозначения поступления вызова по каналу:

```
progressinband=yes|no|never
```

`promiscredir` (для обоих)

Для параметра `promiscredir` могут быть заданы значения `yes` или `no`. Обычно при переадресации звонка на телефон Asterisk использует локальный канал (например, `local/ 18005551212@peer`). Если задан параметр `promiscredir=yes`, Asterisk будет использовать SIP-канал, который позволяет переадресовывать вызовы на удаленные серверы:

```
promiscredir=yes|no
```

Обратите внимание, что, если Asterisk выполняет переадресацию к самой себе, когда `promiscredir=yes`, система получит сообщение `INVITE` со своим Caller ID (ID звонящего) и выявит замыкание на саму себя. SIP не может выполнять замкнутые звонки, поэтому канал будет уничтожен.

`qualify` (равноправный участник)

Для параметра `qualify` может быть задано значение `yes`, `no` или время в миллисекундах. Если задается параметр `qualify=yes`, удаленным равноправным участникам периодически будут посылаться сообщения `NOTIFY` для определения, доступны ли они, и установления величины задержки между ответами. Равноправный участник будет признан недоступным в случае непоступления ответа в течение 2000 мс (изменить это значение по умолчанию можно, задав для параметра `qualify` время ожидания ответа в миллисекундах). Используйте эту опцию в сочетании с `nat=yes`, чтобы поддерживать канал через NAT-устройство активным:

```
qualify=yes|no|количествосекунд
```

`regcontext` (равноправный участник)

Задавая контекст, содержащий некоторые команды, можно сконфигурировать Asterisk на выполнение ряда действий при регистра-

ции равноправного участника сети на вашем сервере. Эта опция используется в сочетании с `regexten`, определяющей, какой добавочный номер должен быть выполнен. Если параметр `regexten` не задан, в качестве добавочного номера используется имя равноправного участника. Asterisk будет динамически создавать и уничтожать для добавочного номера `NoOp` в приоритете 1. Все действия, которые следует выполнять при регистрации, должны начинаться с приоритета 2. Может быть задано несколько параметров `regexten`, разделенных символом `&`. `regcontext` задается для каждого равноправного участника или глобально:

```
regcontext=зарегистрированные_равноправныеучастники
```

`regexten` (равноправный участник)

Опция `regexten` используется в сочетании с `regcontext` для определения добавочного номера, выполняемого в заданном контексте. Если `regexten` не задана явно, в качестве добавочного номера для сопоставления используется имя равноправного участника сети:

```
regexten=1000
```

`rtpholdtimeout` (равноправный участник)

Принимает в качестве аргумента целое число, задается в секундах. Прерывает звонок, если RTP-данные не поступили в течение заданного времени ожидания. Значение `rtpholdtimeout` должно быть больше значения `rtptimeout` (см. также `rtptimeout`):

```
rtpholdtimeout=120
```

`rtpkeepalive` (равноправный участник)

Определяет, как часто Asterisk должна посылать сообщения проверки активности установленного соединения в RTP-потоке, в секундах. Значение по умолчанию – нуль. Это означает, что Asterisk не будет посылать сообщения проверки активности RTP.

```
rtpkeepalive=45
```

`rtptimeout` (равноправный участник)

Принимает в качестве аргумента целое число, соответствующее времени, в секундах, через которое Asterisk прервет вызов в случае непоступления RTP-данных.

```
rtptimeout=60
```

`secret` (для обоих)

Задает пароль, используемый для аутентификации:

```
secret=welcome
```

`setvar` (для обоих)

Задает переменную канала, которая будет доступна с момента создания канала с равноправным участником или пользователем и уничтожена по завершении звонка. Например, чтобы задать переменную канала `foo` со значением `bar`, используйте следующую запись:

```
setvar=foo=bar
```

`username` (равноправный участник)

Поле `username` позволяет выполнять попытки соединения с равноправным участником до того, как он зарегистрировался в вашей системе. При регистрации SIP-устройство сообщает Asterisk, какой SIP URI использовать для связи с ним. Имя пользователя используется в сочетании с `defaultip` для создания SIP URI в заголовке SIP-сообщения `INVITE`. Это может пригодиться для выполнения вызова после перезагрузки. Конечные точки не будут пытаться повторно зарегистрироваться на сервере до истечения срока их регистрации, поэтому вы не будете знать их местоположений. Для нединамических хостов потребуется, чтобы имя пользователя было задано, поскольку оно используется для создания имени пользователя для авторизации:

```
username=john_smith
```

# B

# Справочник по приложениям

Приложения являются основными функциональными элементами диалплана. И приложения, и функции, описанные в приложении F, выполняют некоторые операции с каналом, но функции просто возвращают значения, которые могут быть использованы приложениями. Очень немногие приложения по-прежнему просто возвращают значения, но они, вероятно, будут признаны устаревшими в будущих версиях, их заменят функции диалплана.

Необходимо помнить о некоторых особенностях приложений. Во-первых, они могут завершаться нормально и аварийно. Практически всегда аварийное завершение выполняется, когда приложение выявляет разрыв соединения по каналу (или, если не выявляет, диалплан обнаружит это несколько позже). Приложение также может завершиться аварийно, когда необходимо указать диалплану на то, что некоторое условие не удовлетворено и что необходимо выполнить принудительный разрыв соединения. Во всех остальных случаях приложение будет завершаться нормально, а это свидетельствует о том, что выполнение должно продолжиться в следующем приоритете диалплана.

Часто при необходимости переопределить поведение приложения, обусловливающее разрыв соединения, для него создают оболочку `TryExec()`.

В данном справочнике часто встречается описание *`метка`*. Это сокращенная запись для описания точки диалплана, будь то просто *`приоритет`*, *`добавочный номер`* и *`приоритет`* или *`контекст`*, *`добавочный номер`* и *`приоритет`*. Обратите внимание, что, если *`текстовая метка`* определена для конкретного приоритета, *`приоритет`* может быть заменен этой *`текстовой меткой`* в любом из упомянутых случаев. Больше информации и пример можно найти в описании приложения `GotoIf()`.

Во многих примерах данного приложения используются нумерованные приоритеты, что не рекомендуется при написании диалплана. Мы предпочитаем использовать n для обозначения всех приоритетов, кроме первого (нумерация которого является обязательной), но решили прибегнуть к нумерованным приоритетам, чтобы сделать некоторые примеры более наглядными.

## AddQueueMember()

***Динамически добавляет участников в очередь для заданной очереди вызовов***

AddQueueMember(*имяочереди*[,*интерфейс*[,*приоритет*,[*опция*,[*имяучастника*]]]])

Динамически добавляет заданный *интерфейс* в существующую очередь под именем *имяочереди*, которое определено в файле queues.conf. Если задан, *приоритет* определяет для очередей приоритет данного участника. Участники вызываются в порядке увеличения приоритетов.

По завершении выполнения приложение AddQueueMember() задает переменную канала AQMSTATUS.

Переменной AQMSTATUS будет присвоено одно из следующих значений:

```
ADDED
MEMBERALREADY
NOSUCHQUEUE
```

При вызове AddQueueMember() без аргумента *интерфейс* будет использоваться интерфейс, которым в настоящее время пользуется вызывающий абонент.

Если аргументу *опция* задано значение j, Asterisk не может добавлять интерфейс в указанную очередь и существует приоритет n + 101 (где n – номер текущего приоритета), вызов перейдет в этот приоритет.

Посредством аргумента *имяучастника* может быть задано имя участника очереди. Таким образом, это имя будет использоваться в записях queue_log и событиях интерфейса Asterisk Manager, что упростит идентификацию агента при формировании отчетов:

```
; добавляем SIP/3000 с приоритетом 1 в очередь techsupport
exten => 123,1,AddQueueMember(techsupport,SIP/3000,1)
```

### Смотрите также

Queue(), RemoveQueueMember(), PauseQueueMember(), UnpauseQueueMember(), AgentLogin(), queues.conf

## ADSIProg()

***Загружает ADSI-сценарий в телефон, поддерживающий ADSI***

ADSIProg(*сценарий*)

Программирует телефон, поддерживающий Analog Display Services Interface (ADSI), с помощью заданного сценария. Если сценарий не за-

дан, используется сценарий по умолчанию, asterisk.adsi. Для сценария указывается относительный путь из папки конфигурации Asterisk (обычно это /etc/asterisk/). Также можно указать полный путь к сценарию.

Для получения CPE ID и другой информации от своего ADSI-телефона используйте приложение GetCPEID():

```
; программируем ADSI-телефон с помощью сценария telcordia-1.adsi
exten => 123,1,ADSIProg(telcordia-1.adsi)
```

### Смотрите также

GetCPEID(), adsi.conf

## AgentCallbackLogin()

***Регистрация агента с возможностью обратного вызова***

AgentCallbackLogin([*НомерАгента*][,[*опции*][,[*добавочныйномер*]@*контекст*]])

Разрешает агенту вызовов, идентифицированному параметром *НомерАгента*, регистрироваться в системе очереди вызовов, что позволяет при поступлении вызова для этого агента выполнять обратный вызов к нему.

При поступлении вызова для этого агента Asterisk звонит на заданный добавочный номер (с необязательным контекстом).

Аргумент *опции* может содержать букву s, что означает скрытую авторизацию:

```
; регистрируется в скрытом режиме как агент номер 42
; и определяет, что при поступлении вызова для этого агента
; Asterisk будет звонить на добавочный номер 123
; в контексте internal
exten => 123,1,AgentCallbackLogin(42,s,123@internal)
```

Это приложение является устаревшим, его функциональность замещена логикой диалплана на AEL, размещенного в файле doc/queues-with-callbackmembers.txt в папке исходного кода Asterisk.

### Смотрите также

Queue(), AgentLogin(), AddQueueMember(), RemoveQueueMember(), PauseQueueMember(), UnpauseQueueMember(), AGENT, agents.conf, queues.conf

## AgentLogin()

***Регистрация агента вызовов в системе***

AgentLogin([*НомерАгента*][,*опции*])

Регистрирует текущего вызывающего абонента в системе очереди вызовов как агента обработки вызовов (он может быть идентифицирован

параметром *НомерАгента*). После регистрации агент может принимать вызовы и будет слышать звуковой сигнал в линии при поступлении нового вызова. Агент может прервать текущий вызов, нажав кнопку со звездочкой (*). Если *НомерАгента* не задан, вызывающему абоненту будет предложено ввести свой номер агента. Агенты описаны в файле agents.conf.

Аргумент *опции* может содержать букву `s`, что означает скрытую авторизацию:

```
; регистрируем в скрытом режиме вызывающего абонента как
; агента номер 42, как определено в agents.conf
exten => 123,1,AgentLogin(42,s)
```

### Смотрите также

`Queue()`, `AddQueueMember()`, `RemoveQueueMember()`, `PauseQueueMember()`, `UnpauseQueueMember()`, `AGENT`, agents.conf, queues.conf

## AgentMonitorOutgoing()

***Регистрирует исходящие вызовы агента***

`AgentMonitorOutgoing([`*опции*`])`

Регистрирует все исходящие вызовы, производимые агентом обработки вызовов.

Это приложение пытается выяснить ID агента, выполняющего исходящий вызов, на основании сравнения Caller ID (ID звонящего) текущего интерфейса и глобальной переменной, заданной приложением `AgentCallbackLogin()`. Таким образом, оно должно использоваться только в сочетании (и после него!) с приложением `AgentCallbackLogin()`. Для записи вызовов оно использует не приложение `Monitor()`, а функции для записи разговоров модуля `chan_agent`. Это означает, что процедура записи вызовов должна быть правильно сконфигурирована в файле agents.conf.

По умолчанию записанные звонки сохраняются в папке /var/spool/asterisk/monitor/. Это можно переопределить с помощью параметра `savecallsin` в файле agents.conf.

Если Caller ID и/или ID агента на найден, это приложение передаст управление приоритету `n` + 1, если он существует (где `n` – текущий приоритет).

Если это не переопределено какой-либо опцией, возвращается `0`.

Аргумент опции может включать одно или более из следующих значений:

`d`

В случае ошибки и отсутствия добавочного номера `n` + 101 приложение возвращает `-1`.

`c`

Меняет Call Detail Record (Запись параметров вызова) таким образом, что источник вызова записывается как `Агент/id_агента`.

n

Не формирует предупреждения, если отсутствует Caller ID или ID агента неизвестен. Эта опция полезна, если вы желаете использовать один контекст для звонков агентов и не-агентов.

```
; записываем исходящие звонки для этого агента и меняем
; CDR, чтобы отразить то, что звонок выполняется агентом
exten => 123,1,AgentMonitorOutgoing(c)
```

### Смотрите также

AgentCallbackLogin(), agents.conf

## AGI()

***Выполняет совместимое с AGI приложение***

[E]AGI(*программа*[,*аргументы*])

Выполняет для текущего канала совместимую с Asterisk Gateway Interface программу. AGI-программы обеспечивают возможность внешним программам (которые могут быть написаны практически на любом языке программирования) управлять каналом связи путем воспроизведения аудиофайлов, чтения DTMF-сигналов и т. д. Asterisk обменивается информацией с AGI-программой с помощью стандартных потоков ввода вывода STDIN и STDOUT. Заданные аргументы передаются в AGI-программу.

В качестве программы должен быть задан исполняемый файл из базовой файловой системы. Путь к программе должен вести в папку AGI Asterisk, по умолчанию это /var/lib/asterisk/agi-bin/.

Если необходимо выполнить AGI, когда не существует ни одного канала (как для добавочного номера h), используйте приложение DeadAGI(). Если требуется выполнять AGI удаленно, используется приложение FastAGI().

Если вы хотите выполнять доступ к входящему аудиопотоку из своей AGI-программы, вместо AGI() используйте приложение EAGI(). Тогда входящий аудиопоток может читаться в дескриптор файла 3.

Если происходит преждевременный разрыв соединения, процессу, запущенному командой AGI, будет послан сигнал HUP, извещающий о завершении соединения. Если ваша программа не перехватит этот сигнал, она будет завершена. Это поведение можно переопределить, задав для переменной канала AGISIGHUP значение 0:

```
; вызываем демонстрационную AGI-программу
exten => 123,1,AGI(agi-test)
exten => 123,2,EAGI(eagi-test)
```

### Смотрите также

DeadAGI(), FastAGI(), главу 9

# AlarmReceiver()

***Предоставляет поддержку для получения сигналов с панели охранной или пожарной сигнализации***

`AlarmReceiver()`

Эмулирует приемник сигналов тревоги и позволяет Asterisk принимать и декодировать специальные данные панелей пожарной и/или охранной сигнализации. На данный момент поддерживается только формат Ademco Contact ID.

Будучи вызванным, приложение `AlarmReceiver()` подтвердит установление связи с панелью сигнализации, будет принимать события, проверять их достоверность, подтверждать их и сохранять до тех пор, пока панель не разорвет соединение. Как только панель разорвет соединение, приложение будет выполнять строку команд, заданную настройкой `eventcmd` в файле alarmreceiver.conf, и передавать события на стандартный ввод приложения. Файл alarmreceiver.conf также содержит настройки синхронизации DTMF и громкости тонов подтверждения приема.

```
; настраиваем Asterisk, чтобы она отвечала на вызовы
; поддерживаемой панели пожарной сигнализации
exten => s,1,AlarmReceiver()
```

Надежность этого приложения не гарантируется, поэтому не полагайтесь на него без всестороннего тестирования. Используя это приложение без тестирования, вы подвергаете свою жизнь и собственность большой опасности.

## Смотрите также

alarmreceiver.conf

# AMD()

***Выявление автоответчика***

`AMD([initialSilence[,greeting[,afterGreetingSilence[,totalAnalysisTime[,minimumWordLength[,betweenWordsSilence[,maximumNumberOfWords[,silenceThreshold]]]]]]])`

Это приложение пытается установить наличие автоответчика на основании шаблонов синхронизации. Это приложение обычно используется исходящими вызовами, берущими начало или в файлах вызовов, или в интерфейсе Asterisk Manager. Приложение сообщает, какой тип вызова был выявлен, задавая переменной `AMDSTATUS` одно из следующих значений:

`MACHINE` (машина)

Считается, что вызываемая сторона является автоответчиком.

`HUMAN` (человек)

Считается, что вызываемая сторона является человеком, а не автоответчиком.

NOTSURE (не уверен)

Приложение не смогло определить, является ли вызываемая сторона человеком или автоответчиком.

HANGUP (разрыв)

В процессе определения произошел разрыв соединения.

Приложение AMD() также указывает в переменной канала AMDCAUSE причину, на основании которой делается заключение, обозначенное в переменной AMDSTATUS. Для переменной AMDCAUSE будет задано одно из следующих значений:

TOOLONG-*общее_время*
INITIALSILENCE-*продолжительность_паузы-начальная_пауза*
HUMAN-*продолжительность_паузы-пауза_после_приветствия*
MAXWORDS-*количество_слов-максимальное_число_слов*
LONGGREETING-*продолжительность_разговора-приветствие*

Все параметры данного приложения помогают настроить его так, чтобы оно могло более эффективно оценивать разницу между человеком и автоответчиком. Если параметры не переданы в это приложение, Asterisk прочитает значения по умолчанию, описанные в amd.conf. К этим параметрам относятся:

*initialSilence* (начальная пауза)

Максимальная продолжительность паузы перед приветствием. Если это значение превышено, для переменной AMDSTATUS будет задано значение MACHINE.

*greeting* (приветствие)

Максимальная продолжительность приветствия. Если превышена, для переменной AMDSTATUS будет задано значение MACHINE.

*afterGreetingSilence* (пауза после приветствия)

Максимальная пауза после обнаружения приветствия. Если превышена, для переменной AMDSTATUS будет задано значение MACHINE.

*totalAnalysisTime* (общее время анализа)

Максимальное время, предоставляемое алгоритму для принятия решения о том, является ли вызываемая сторона человеком или автоответчиком.

*minimumWordLength* (минимальная длина слова)

Если продолжительность разговора короче, чем minimumWordLength, это не будет считаться речью человека.

*betweenWordsSilence* (пауза между словами)

Минимальная пауза после слова, чтобы считать следующий аудиосигнал новым словом.

*maximumNumberOfWords* (максимальное число слов)

Максимальное число слов в приветствии. Если это значение превышено, для переменной AMDSTATUS будет задано значение MACHINE.

*silenceThreshold* (пороговая продолжительность паузы)

Чувствительность алгоритма при выявлении паузы.

```
; Используем алгоритм выявления автоответчика. Если
; вызываемая сторона является человеком, устанавливаем
; соединение с Бобом. В противном случае воспроизводим
; сообщение и разрываем соединение
exten => 123,1,Answer()
exten => 123,n,AMD()
exten => 123,n,GotoIf($["${AMDSTATUS}" = "HUMAN"]?human:machine)
exten => 123,n(machine),WaitForSilence(2000)
exten => 123,n,Playback(asterisk-friend)
exten => 123,n,Hangup()
exten => 123,n(human),Verbose(3, We've got a human on the line!)
exten => 123,n,Playback(transfer)
exten => 123,n,Dial(SIP/bob)
exten => 123,n,Playback(im-sorry)
exten => 123,n,Hangup()
```

### Смотрите также

`WaitForSilence()`

## Answer()

***Устанавливает соединение, если по каналу поступает вызов***

`Answer([`*задержка*`])`

Обусловливает установление соединения Asterisk с каналом, если по нему в настоящий момент поступает вызов. Если по текущему каналу вызов не производится, это приложение ничего не делает.

Если задана задержка, Asterisk ответит на вызов и перейдет к следующему приоритету диалплана только по прошествии заданного количества миллисекунд.

Если нет веских оснований не делать этого, `Answer()` рекомендуется использовать для канала перед вызовом всех остальных приложений. Существует несколько ключевых приложений, которые требуют, чтобы перед их выполнением было выполнено приложение `Answer()`. В противном случае они могут работать некорректно:

```
exten => 123,1,Answer(750)
exten => 123,n,Playback(tt-weasels)
```

### Смотрите также

`Hangup()`

## AppendCDRUserField()

***Добавляет значение в поле пользователя записи Call Detail Record***

`AppendCDRUserField(`*значение*`)`

Добавляет значение в поле пользователя записи Call Detail Record (CDR). Поле пользователя часто используется для хранения произ-

вольных данных о вызове, которые не подходят для всех остальных полей:

```
; задаем в поле пользователя значение 'abcde'
exten => 123,1,SetCDRUserField(abcde)
; теперь добавлям в конец 'xyz'
exten => 123,1,AppendCDRUserField(xyz)
```

Это приложение было признано устаревшим и заменено CDR-функцией.

```
exten => 123,1,Set(CDR(userfield)=${CDR(userfield)}12345)
```

## Смотрите также

`SetCDRUserField()`, `ForkCDR()`, `NoCDR()`, `ResetCDR()`, CDR

# Authenticate()

***Требует от вызывающего абонента введения правильного пароля для продолжения выполнения***

`Authenticate(`*пароль*`[,`*опции*`[,`*максимумсимволов*`]])`

Требует от вызывающего абонента ввести заданный пароль, чтобы продолжать выполнение следующего приоритета диалплана. `Authenticate()` дает вызывающему абоненту три попытки для правильного введения пароля. Если за эти три попытки вызывающий абонент так и не смог ввести правильный пароль, выполняется разрыв соединения.

Если пароль начинается с символа /, он трактуется как файл, содержащий список действительных паролей (по одному в строке). Пароли также могут храниться в базе данных Asterisk (AstDB); см. опцию `d` ниже.

Параметр *максимумсимволов* задает максимальное число символов, которое может ввести вызывающий абонент. Если этот параметр не задан, приложение будет принимать неограниченное число символов и станет ожидать от вызывающего абонента нажатия кнопки # после введения кода аутентификации.

Может быть задана одна или несколько опций из следующего списка:

`a`

Присваивает CDR-полю `accountcode`, а переменной канала `ACCOUNTCODE` — значение введенного пароля.

`d`

Трактует путь не как файл, а как ключ базы данных Asterisk, в которой следует искать пароль. При использовании ключа базы данных значением, ассоциированным с ключом, может быть все что угодно.

`j`

Поддерживает переход к приоритету `n` + 101 в случае неудачной аутентификации.

`m`

Трактует заданный путь как файл, содержащий список кодов учетных записей и хешей паролей, разделенных символом `:` (двоеточие), по одному в строке. При совпадении одного из паролей для канала будет задан код учетной записи, соответствующий коду, указанному в файле.

`r`

Удаляет ключ базы данных после успешного входа (действительна только с опцией `d`).

```
; вынуждаем вызывающего абонента ввести пароль,
; прежде чем выполнять остальные действия,
; и сохраняем введенный пароль в CDR-поле 'accountcode'
exten => 123,1,Answer()
exten => 123,n,Authenticate(1234,a)
exten => 123,n,Playback(pin-number-accepted)
exten => 123,n,SayDigits(${ACCOUNTCODE})
```

### Смотрите также

`VMAuthenticate()`, `DISA()`, главу 6

## Background()

***Воспроизводит файл, принимая при этом сигналы тонального набора (DTMF)***

`Background(`*имяфайла1*`[&`*имяфайла2*`...][,`*опции*`[,`*язык*`]])`

Воспроизводит заданные аудиофайлы в процессе ожидания введения пользователем DTMF-кодов. Как только пользователь начинает вводить DTMF-коды, воспроизведение прекращается. Asterisk пытается найти соответствующий добавочный номер в целевом контексте (или текущем контексте, если целевой контекст не задан), и, как только будет найдено однозначное соответствие, выполнение диалплана продолжится в соответствующем добавочном номере.

*имяфайла* должно быть задано без расширения файла, поскольку Asterisk автоматически выбирает формат файла с минимальными затратами на преобразование.

Допустимыми опциями являются следующие:

`s`

Сообщение воспроизводиться не будет, если канал находится не в состоянии «соединен» (то есть еще не получен ответ на вызов). Если `s` задана, приложение будет возвращаться сразу же, как только канал окажется не в состоянии «соединен».

`n`

Не отвечать на вызов до воспроизведения заданного файла. Без этой опции ответ на вызов будет произведен автоматически перед вос-

произведением звука. Не все каналы поддерживают воспроизведение сообщений до ответа на вызов.

`m`

Прерывать воспроизведение, только если введенный код соответствует одноразрядному добавочному номеру в целевом контексте.

Аргумент *язык* может использоваться для задания языка воспроизводимого приглашения, если он отличается от используемого языка канала.

```
exten => 123,1,Answer()
exten => 123,2,Background('exter-ext-of-person');
```

### Смотрите также

`ControlPlayback()`, `WaitExten()`, `BackgroundDetect()`, `TIMEOUT`

## BackgroundDetect()

***Воспроизводит файл в фоновом режиме и выявляет разговор***

`BackgroundDetect(`*имяфайла*`[,`*sil*`[,`*min*`[,`*max*`]]])`

Аналогично `Background()`, но пытается выявить разговор.

Во время воспроизведения файла выполняется отслеживание аудиосигналов во входящем потоке. Если период отсутствия тишины длится больше *min* миллисекунд, но еще меньше *max* миллисекунд и за ним следует пауза продолжительностью как минимум *sil* миллисекунд, воспроизведение звука прерывается и выполнение переходит в добавочный номер `talk` (разговор), если таковой доступен.

Если не заданы, параметры *sil*, *min* и *max* по умолчанию принимают значения 1000 мс, 100 мс и бесконечное количество соответственно.

```
exten => 123,1,BackgroundDetect(tt-monkeys)
exten => 123,2,Playback(im-sorry)
exten => talk,1,Playback(yes-dear)
```

### Смотрите также

`Playback()`, `Background()`

## Busy()

***Обозначает состояние занятости канала***

`Busy([`*времяожидания*`])`

Указывает каналу обозначить состояние «занято», а затем ожидает, когда пользователь повесит трубку или разрыва соединения по истечении времени ожидания (заданного необязательным параметром *времяожидания* в секундах).

Это приложение сигнализирует о состоянии занятости только для соединенных каналов. У каждого типа каналов имеется собственный

способ оповещения вызывающего абонента о состоянии занятости. Можно использовать Playtones(busy) для воспроизведения сигнала «занято».

```
exten => 123,1,Playback(im-sorry)
exten => 123,2,Playtones(busy)
exten => 123,3,Busy()
```

### Смотрите также

Congestion(), Progress(), Playtones(), Hangup()

## ChangeMonitor()

***Меняет имя файла для записи разговора по каналу***

ChangeMonitor(*базовое_имяфайла*)

Меняет имя записанного файла для канала, созданного приложением Monitor(). Это приложение не оказывает никакого эффекта, если на канале не производится запись разговора. Аргумент *базовое_имяфайла* – это новое базовое имя файла, которое должно использоваться для записи разговора в канале.

```
; начинаем запись разговора в данном канале,
; используя базовое имя файла 'sample'
exten => 123,1,Monitor(sample)
; меняем базовое имя файла на 'example'
exten => 123,2,ChangeMonitor(example)
```

### Смотрите также

Monitor(), StopMonitor(), MixMonitor()

## ChanIsAvail()

***Определяет, доступен ли в настоящее время заданный канал***

ChanIsAvail(*технология1/ресурс1*[&*технологя2/ресурс2...*][,*опции*])

Проводит проверку на определение доступности какого-либо из запрашиваемых каналов. Это приложение также задает следующие переменные каналов:

AVAILCHAN

Имя доступного канала, включая номер сеанса вызова, используемого для проведения проверки.

AVAILORIGCHAN

Каноническое имя канала, используемое для его создания, то есть имя канала без номера сеанса.

*AVAILSTATUS*

Код статуса канала.

Если задана опция s (означающая state – состояние), Asterisk будет считать используемый канал недоступным, даже если он может принять другой вызов.

Если задана опция j и не доступен ни один из запрашиваемых каналов, выполнение переходит в приоритет n + 101 (где n – текущий приоритет), если этот приоритет существует.

```
; проверяет, доступны ли каналы Zap/1 и Zap/2
exten => 123,1,ChanIsAvail(Zap/1&Zap/2)
; выводит имя доступного канала в CLI Asterisk
exten => 123,2,Verbose(0,${AVAILORIGCHAN})
```

Это приложение работает некорректно для каналов MGCP.

## ChannelRedirect()

***Перенаправляет канал в новую точку диалплана***

ChannelRedirect(*канал*,[[*контекст*,]*добавочныйномер*,]*приоритет*)

Это приложение перенаправляет заданный канал в новый приоритет диалплана. Если *добавочныйномер* не задан, принимается текущий добавочный номер. Если *контекст* не задан, будет принят текущий контекст:

```
; Перенаправляет SIP/Bob к музыке во время ожидания
; при наборе добавочного номера 123
exten => 123,1,ChannelRedirect(SIP/Bob,124,1)
exten => 124,1,Answer()
exten => 124,2,MusicOnHold()
```

### Смотрите также

Transfer()

## ChanSpy()

***Слушает разговор в канале и может посылать сигналы в вызывающий канал***

ChanSpy([*префикскanала*[,*опции*]])

Это приложение используется для прослушивания аудиосигнала, проходящего по каналу Asterisk в обоих направлениях. Если задан параметр *префикскanала*, прослушиваться будут только каналы, начинающиеся с этого префикса.

При прослушивании канала могут осуществляться следующие действия:

- Набор # циклически меняет уровень громкости.
- Набор * заставит приложение перейти к прослушиванию следующего доступного канала.

- Набор последовательности символов, завершенный нажатием кнопки #, создает имя канала (которое будет добавлено после `префикс_канала`). Например, если запустить `ChanSpy(Zap)`, а затем в процессе прослушивания набрать символы 42#, начнется прослушивание канала Zap/42.

Параметр *опции* может содержать нуль или более следующих опций:

`b`

Прослушивать только соединенные каналы.

`g(`*группа*`)`

Прослушивать только каналы с переменной канала `SPYGROUP`, в которой в необязательном списке с разделяющими двоеточиями должна быть указана группа.

`q`

Скрытый режим. Указывает приложению, начиная прослушивание, не подавать звуковой сигнал или не читать имя выбранного канала.

`r[(`*базовоеимя*`)]`

Записывает разговор, ведущийся по каналу, в папку очереди для записей разговоров (обычно это /var/spool/asterisk/monitor). Необязательное *базовоеимя* задает базовое имя файла для записей, значение по умолчанию – `chanspy`.

`v([`*значение*`])`

Настраивает громкость прослушиваемого аудиосигнала. Значение должно быть в диапазоне от 4 до –4. Отрицательное значение сделает звук тише, тогда как положительное – громче.

`w`

Режим шепота. Позволяет прослушивающему каналу «говорить» с прослушиваемым каналом. При этом ни один другой соединенный канал не сможет слышать этот разговор.

`W`

Закрытый режим шепота. Позволяет прослушивающему каналу «говорить» с прослушиваемым каналом без возможности слышать аудиосигнал из прослушиваемого канала.

```
; Прослушиваем каналы Zap в режиме шепота
exten => 123,1,ChannelSpy(Zap,w)
```

## Смотрите также

`ExtenSpy()`

## Congestion()

***Устанавливает состояние перегрузки канала***

Congestion([*времяожидания*])

Указывает каналу индицировать перегрузку, а затем ожидает, когда пользователь повесит трубку или разрыва соединения по истечении времени ожидания (заданного необязательным параметром *времяожидания* в секундах).

Это приложение сигнализирует о перегрузке только на дальний конец соединения; оно фактически не воспроизводит тональный сигнал перегрузки линии абоненту. Для воспроизведения сигнала перегрузки используйте приложение Playtones(congestion).

При использовании данной команды без времени ожидания возникает риск того, что канал задержится в этом состоянии. В этом нет необходимости, когда вы просто хотите проинформировать пользователя о перегруженности канала. Используйте приложение Playtones(congestion), чтобы абонент услышал сигнал «занято» повышенной частоты, а затем выполните Hangup().

Всегда завершается аварийно:

```
; для Caller ID 555-1234 всегда воспроизводить
; сигнал перегруженности линии
exten => 123,1,GotoIf($[${CALLERID(num)} = 5551234]?5:2)
exten => 123,2,Playtones(congestion)
exten => 123,3,Congestion(3)
exten => 123,4,Hangup()
exten => 123,5,Dial(Zap/1)
```

### Смотрите также

Busy(), Progress(), Playtones(), Hangup()

## ContinueWhile()

***Выполняет перезапуск цикла While()***

ContinueWhile()

Возвращается к началу цикла While() и повторно вычисляет условное выражение.

### Смотрите также

While(), ExitWhile()

## ControlPlayback()

***Воспроизводит файл с возможностью ускоренной перемотки вперед и назад***

ControlPlayback(*файл*[,*skipms*[,*ff*[,*rew*[,*stop*[,*pause*[,*restart*[,*опции*]]]]]]])

Воспроизводит указанный файл (без расширения файла), предоставляя при этом пользователю возможность перемещаться по файлу вперед и назад, нажимая кнопки ff и rew. По умолчанию можно использовать кнопки * и # для перемотки файла назад и вперед соответственно.

Опция *skipms* определяет, на сколько секунд записи вперед или назад выполняется переход при каждом нажатии кнопки ff или rew.

Если задан аргумент *stop*, приложение будет останавливать воспроизведение при нажатии кнопки stop.

Также может быть задан аргумент *pause*, который определяет, что при нажатии кнопки pause воспроизведение файла будет приостановлено. При повторном нажатии кнопки pause воспроизведение файла возобновится.

Если задан параметр *restart*, то при нажатии определенной кнопки воспроизведение файла будет начато сначала.

Если для параметра *опции* задано значение j и указанного файла не существует, приложение переходит в приоритет n + 101, если таковой присутствует (где n – номер текущего приоритета).

Приложение ControlPlayback() по завершении выполнения задает значение для переменной канала CPLAYBACKSTATUS. Переменной CPLAYBACK STATUS будет присвоено одно из следующих значений:

SUCCESS (успех)
USERSTOPPED (остановлен пользователем)
ERROR (ошибка)

```
; позволяет вызывающему абоненту управлять
; воспроизведением этого файла
exten => 123,1,ControlPlayback(tt-monkeys|3000|#|*|5|0)
```

### Смотрите также

Playback(), Background(), Dictate(),

## DateTime()

***Проговаривает дату и/или время в указанном пользователем формате***

DateTime([*unixtime*[,*часовойпояс*[,*формат*]]])

Если задан параметр *unixtime*, приложение проговаривает эти дату и время. В противном случае оно проговаривает текущие дату и время. Если задан *часовойпояс*, дата и время вычисляются соответственно этому часовому поясу. В противном случае используется зональная настройка сервера Asterisk. Если задан параметр *формат*, дата и время будут проговариваться соответственно этому формату. (Подробнее о формате даты и времени можно узнать из образца файла voicemail.conf.)

```
; проговаривает текущую дату и время
; в нескольких часовых поясах
exten => 123,1,DateTime(,America/New_York)
exten => 123,2,DateTime(,America/Chicago)
exten => 123,3,DateTime(,America/Denver)
exten => 123,4,DateTime(,America/Los_Angeles)
```

## DBdel()

***Удаляет ключ из AstDB***

DBdel(*семейство/ключ*)

Удаляет ключ, заданный параметром *ключ*, из семейства ключей *семейство* в AstDB.

```
exten => 123,1,DBput(test/name=John) ; добавляем имя в AstDB
exten => 123,2,DBget(NAME=test/name) ; извлекаем имя из AstDB
exten => 123,3,DBdel(test/name)      ; удаляем из AstDB
```

Это приложение является устаревшим и заменено функцией DB_DELETE().

### Смотрите также

DB_DELETE(), DBdeltree(), DB

## Dbdeltree()

***Удаляет семейство или дерево ключей из AstDB***

DBdeltree(*семейство*[*/деревоключей*])

Удаляет заданное *семейство* или *деревоключей* из AstDB.

```
; создает пару записей в AstDB
exten => 123,1,DBput(test/blue)
exten => 123,2,DBput(test/green)
; теперь удаляем семейство ключей test
exten => 123,3,DBdeltree(test)
```

### Смотрите также

DB_DELETE(), DBdel(), DB

## DeadAGI()

***Выполняет AGI-совместимый сценарий для «мертвого» (разъединенного) канала***

DeadAGI(*программа*,*аргументы*)

Выполняет AGI-совместимую программу для «мертвого» (разъединенного) канала. AGI позволяет Asterisk запускать внешние программы, написанные практически на любом языке программирования, для

управления каналом связи, воспроизведения аудиофайлов, чтения DTMF-сигналов и т. д. посредством обмена информацией по AGI-протоколу по STDIN и STDOUT. Аргументы, заданные параметром *аргументы*, будут переданы в программу.

Это приложение было написано специально для «мертвых» каналов, поскольку обычный AGI-интерфейс не работает корректно, если канал разъединен.

Чтобы получить список всех доступных команд AGI, используйте команду интерфейса командной строки `show agi`.

```
exten => h,1,DeadAGI(agi-test)
```

### Смотрите также

`AGI()`, `FastAGI()`

## Dial()

***Пытается соединить каналы***

`Dial(`*технология/имяпользователя*`:`*пароль*`@`*имяхоста*`/`*добавочныйномер*`[&`*технология2/peer2...*`][,`*времяожидания-ответа*`[,`*флаги*`[,`*URL*`]]])`

Позволяет соединять все возможные типы каналов[1]. `Dial()` – самое важное приложение в Asterisk; наверняка вы периодически будете возвращаться к этому разделу.

Для приложения `Dial()` приемлем любой действительный тип каналов (такие, как SIP, IAX2, H.323, MGCP, Local или Zap), но то, какие параметры необходимо будет передать для канала, зависит от информации, которая требуется каналу данного типа для работы. Например, SIP-каналу для соединения понадобится сетевой адрес и пользователь, тогда как Zap-канал потребует телефонный номер.

Когда задается сетевой тип канала, в качестве опций `Dial()` могут передаваться хост назначения (имя или IP-адрес), имя пользователя, пароль и удаленный добавочный номер или можно указать имя записи канала в соответствующем файле .conf; тогда вся необходимая информация должна поступать из этого файла. Имя пользователя и пароль можно заменить именем, указанным в квадратных скобках (`[]`) в конфигурационном файле канала. Имя хоста является необязательным.

Вот действительное выражение `Dial`:

```
exten => s,1,Dial(SIP/sake:arigato@thathostoverthere.tld)
```

[1] Тот факт, что Asterisk может успешно соединять IAX, SIP, H.323, Skinny, PRI, FX(O/S) и все что угодно, поразителен, но, вероятно, самое удивительное – это локальный (Local) канал. Благодаря тому что одной командой `Dial()` можно соединять множество локальных каналов, событие `Dial()` может запускать множество совершенно независимых и уникальных действий в других частях диалплана. Данная концепция поистине революционная, и ее надо испытать, чтобы поверить в это.

Такая запись аналогична предыдущей:

```
exten => s,1,Dial(SIP/some_SIP_friend)
```

но будет работать, только если в sip.conf есть канал `[some_SIP_friend]`, описание которого включает `fromuser=sake`, `password=arigato` и `host=thathostoverthere.tld`.

Часто добавочный номер указывается после адреса:

```
exten => s,1,Dial(IAX2/user:pass@otherend.com/500)
```

Таким образом мы указываем дальнему концу соединения направить вызов на добавочный номер 500 в том же контексте, в который поступил вызов. Необязательно указывать добавочный номер в `Dial()`, поскольку может использоваться информация конфигурационного файла канала удаленного конца соединения или удаленный сервер передаст вызов на добавочный номер `s` в контексте, в котором поступил вызов. В конечном счете, дальний конец соединения управляет тем, что происходит с вызовом; вы можете только запрашивать определенную обработку.

Если не задан параметр *время_ожидания_ответа*, канал будет совершать вызов неопределенно долго. Это не всегда плохо, поэтому не надо полагать, что это обязательный параметр; просто знайте, что «неопределенно долго» может означать «очень долго». Параметр *время_ожидания_ответа* задается в секундах и всегда следует за адресной информацией:

```
exten => s,1,Dial(IAX2/user:pass@otherend.com/500,время_ожидания_ответа)
```

Мощь приложения `Dial()` во многом определяется флагами. Они указываются после адреса и времени ожидания:

```
exten => s,1,Dial(IAX2/user:pass@otherend.com/500,60,флаги)
```

Если время ожидания не задано и вы желаете задать флаги, необходимо обозначить место, где можно указать время ожидания. Делается это введением дополнительной запятой на месте времени ожидания, как здесь:

```
exten => s,1,Dial(IAX2/user:pass@otherend.com/500,,флаги)
```

Для приложения `Dial()` могут использоваться следующие флаги:

`A( x )`

Воспроизводится приветствие для вызываемой стороны; *x* – имя звукового файла, используемого в качестве приветствия.

`C`

Выполняется сброс записи Call Detail Record для звонка. Поскольку в CDR выставляется время ответа (`Answer()`) на звонок, вероятно, вы захотите сбросить запись CDR, чтобы пользователю не приходилось оплачивать время, прошедшее до вызова приложения `Dial()`.

`d`

Пользователь может набирать одноразрядный добавочный номер в процессе ожидания ответа на звонок. Тогда звонок перейдет на

этот добавочный номер (или в текущем контексте, или, если производится выход, в контексте, заданном переменной EXITCONTEXT).

D([ *вызываемый* ][: *вызывающий* ])

DTMF-коды передаются после ответа на звонок, но до того, как выполнено соединение. Параметр *вызываемый* передается вызываемой стороне, а параметр *вызывающий* – вызывающей стороне. Эти параметры могут использоваться по отдельности.

f

Caller ID (ID звонящего) вызывающей стороны принудительно устанавливается как добавочный номер, связанный с каналом, с помощью подсказки диалплана. Часто это используется, когда провайдер разрешает задавать в качестве Caller ID только выделенные для вас номера. Например, при наличии PRI флаг f использовался бы для переопределения любого заданного локально Caller ID для SIP-телефона.

g

Выполнение диалплана продолжается в текущем контексте, если вызываемый абонент вешает трубку.

G( *контекст* ^ *добавочныйномер* ^ *приоритет* )

Когда ответ на звонок получен, вызывающая сторона переводится в заданный приоритет, а вызываемая сторона – в заданный приоритет + 1. В сочетании с этой опцией не могут использоваться никакие дополнительные опции, определяющие операции после ответа.

h

Позволяет вызванному абоненту разорвать соединение нажатием кнопки *.

H

Позволяет вызывающему абоненту разорвать соединение нажатием кнопки *.

i

Asterisk будет игнорировать все попытки перенаправить вызов для данной попытки вызова.

j

Asterisk будет переходить в приоритет n + 101 в случае занятости запрашиваемых каналов (где n – текущий приоритет).

L( *x* [: *y* ][: *z* ])

Продолжительность звонка ограничивается *x* миллисекундами, дается предупреждение, когда осталось *y* миллисекунд, и предупреждение повторяется каждые *y* миллисекунд вплоть до истечения допустимой продолжительности разговора. Параметр *x* – обязатель-

ный; *y* и *z* – необязательные. Для обеспечения дополнительного контроля могут использоваться следующие специальные переменные:

LIMIT_PLAYAUDIO_CALLER=yes|no

Определяет, воспроизводится ли звуковой файл для вызывающего абонента. По умолчанию – yes.

LIMIT_PLAYAUDIO_CALLEE=yes|no

Определяет, воспроизводится ли звуковой файл для вызываемого абонента.

LIMIT_TIMEOUT_FILE= *имяфайла*

Определяет, какой файл воспроизводится, когда время истекло.

LIMIT_CONNECT_FILE= *имяфайла*

Определяет, какой файл воспроизводится в начале вызова.

LIMIT_WARNING_FILE= *имяфайла*

Определяет, какой файл воспроизводится, если определен аргумент *y*. По умолчанию проговаривается оставшееся время.

m[*класс*]

Музыка для вызывающей стороны проигрывается до тех пор, пока не будет получен ответ на звонок. Можно также указать (необязательно) класс музыки во время ожидания, заданный в файле musiconhold.conf.

M( *x* [ ^*аргумент* ])

При соединении выполняется макрос *x* и передаются (необязательно) аргументы, разделенные символом ^. Макрос также может задавать одно из приведенных ниже значение для переменной канала MACRO_RESULT, чтобы обозначить, что должно произойти после завершения макроса:

ABORT (прервать)

Повесить трубку на обоих концах соединения.

CONGESTION (перегрузка)

Действовать так, как при перегрузке линии.

BUSY (занято)

Действовать так, как при занятости линии. Если задана опция j, вызов перенаправляется в приоритет n + 101, где n – текущий приоритет.

CONTINUE (продолжить)

Отключить вызываемую сторону и продолжить выполнение диалплана.

GOTO:<контекст>^<добавочныйномер>^<приоритет>

Выполнить переадресацию вызова в заданную точку.

В сочетании с этой опцией нельзя использовать никакие другие дополнительные опции, определяющие действия после ответа. Также сервисы офисной АТС не выполняются для вызванного канала, поэтому не получится задать значения времени ожидания посредством функции TIMEOUT в этом макросе.

n

Эта опция является модификатором для экранного/конфиденциального режима (screen/privacy mode). Она определяет, что вступления (introductions) не должны сохраняться в папке priv-callerintros.

N

Эта опция является модификатором для экранного/конфиденциального режима. Она указывает Asterisk не экранировать вызов, если представлен Caller ID.

o

Использует Caller ID, полученный по входящему плечу вызова, в качестве Caller ID на исходящем плече вызова. Это полезно, если вы принимаете вызов и перенаправляете его в другую точку, но хотите передать Caller ID из входящего плеча вызова, а не заменять его локальным Caller ID. Таким было поведение по умолчанию Asterisk в версиях до 1.0.

O[*x*]

Эта опция включает режим услуг оператора (Operator Services) для канала Zaptel. При использовании в не-Zaptel-интерфейсе эта опция будет проигнорирована. После ответа вызываемой стороны (предположительно, станции услуг оператора) источник вызова теряет контроль над линией. Вызывающий абонент может повесить трубку, но линия не будет освобождена да тех пор, пока не повесит трубку вызываемая сторона (оператор). Если аргумент не задан или задана 1, то, когда вызывающая сторона вешает трубку, ее телефон немедленно зазвонит. Если задан аргумент 2, то, когда «оператор» выполняет мгновенный сброс магистральной линии, зазвонит телефон вызывающего абонента.

p

Активирует режим экранирования. По сути, это конфиденциальный режим без запоминания.

P[(*x*)]

Задает конфиденциальный режим. Может быть задан необязательный параметр *x* как значение семейство/ключ локальной базы данных AstDB. Эта опция полезна для принятия звонков на основании черного списка (явно запрещающего звонки с указанных номеров) или белого списка (явно перечисляющего номера, звонки с которых должны приниматься обязательно). См. также LookupBlacklist().

r

Указывает генерировать сигналы вызова вызывающему абоненту, не передавая никаких аудиосигналов до тех пор, пока не получен ответ на звонок. Обычно этот флаг не нужен, поскольку Asterisk сама будет генерировать сигналы вызова.

S(*x*)

Прерывает звонок через *x* секунд после ответа вызываемой стороны.

t

Разрешает вызываемой стороне переадресовать звонок, нажав кнопку #. Пожалуйста, обратите внимание, что, если используется эта опция, повторные приглашения выключены, поскольку Asterisk необходимо отслеживать вызов, чтобы зафиксировать момент нажатия кнопки # вызываемой стороной.

T

Разрешает ответившему абоненту переадресовывать соединенный вызов, нажав кнопку #. Опять же, обратите внимание, что при использовании этой опции повторные приглашения выключены, поскольку Asterisk необходимо отслеживать вызов, чтобы зафиксировать момент нажатия кнопки # вызываемой стороной.

w

Разрешает ответившему абоненту начинать и останавливать запись разговора на диск, нажав последовательность кнопок `automon` (которая задана в файле features.conf). Если задана переменная `TOUCH_MONITOR`, ее значение будет передано как аргумент в приложение `Monitor()` в начале записи. Если переменная не задана, в `Monitor()` передаются значения по умолчанию, `WAV||m`.

W

Позволяет вызывающему абоненту записывать разговор на диск, нажав последовательность кнопок `automon` (которая задана в файле features.conf).

k

Позволяет ответившему абоненту парковать вызов (переключать на другой телефон), передав DTMF-последовательность, определенную для парковки вызовов в файле features.conf.

K

Позволяет вызывающей стороне парковать вызов, передав DTMF-последовательность, определенную для парковки вызовов в файле features.conf.

Если включен аргумент *URL*, этот URL будет отправлен каналу (если поддерживается).

Если задать переменную канала OUTBOUND_GROUP перед вызовом Dial(), все каналы типа peer, созданные данным приложением, будут помещаться в эту группу вызовов. В следующем примере все каналы peer, созданные приложением Dial(), будут частью группы вызовов test:

```
; используем OUTBOUND_GROUP
exten => 123,1,Set(OUTBOUND_GROUP=test)
exten => 123,n,Dial(IAX2/anotherbox/12345)
```

Если задана переменная OUTBOUND_GROUP_ONCE, все каналы peer, созданные данным приложением, будут помещены в эту группу. Однако, в отличие от OUTBOUND_GROUP, эта переменная будет сброшена после использования.

Приложение Dial() при выходе задает следующие переменные:

DIALEDTIME (время дозвона)

Общее время с момента начала выполнения Dial() до завершения.

ANSWEREDTIME (время ответа)

Общее время вызова.

DIALSTATUS (статус звонка)

Статус звонка, задается одним из следующих значений:

CHANUNAVAIL (канал недоступен)

Канал недоступен.

CONGESTION (перегрузка)

Канал возвратил сигнал перегрузки, обычно свидетельствующий о невозможности завершить соединение.

NOANSWER (не отвечает)

Канал не ответил в течение времени, заданного опцией *времяожидания-ответа*.

BUSY (занят)

Вызываемый канал в настоящее время занят.

ANSWER (ответ)

Канал ответил на вызов.

CANCEL (отмена)

Вызов был отменен.

DONTCALL (не вызывать)

Вызов был переведен в состояние DONTCALL опциями экранирования или конфиденциальности.

TORTURE (отключение)

Вызов был переведен в состояние TORTURE опциями экранирования или конфиденциальности.

INVALIDARGS (недействительные аргументы)

В приложение Dial() были переданы недействительные аргументы.

```
; набираем семизначный номер по Zap-каналу 4
exten => 123,1,Dial(Zap/4/2317154)

; набираем тот же номер, но на этот раз дозвон будет
; осуществляться только в течение 10 с,
; после чего будет продолжено выполнение диалплана
exten => 124,1,Dial(Zap/4/2317154,10)
exten => 124,2,Playback(im-sorry)
exten => 124,3,Hangup()

; набираем тот же номер, но на этот раз без времени
; ожидания, и используем флаги t, T и m
exten => 125,1,Dial(Zap/4/2317154,,tTm)

; набираем добавочный номер 500 на удаленном хосте
; (по протоколу IAX), используя заданные
; имя пользователя и пароль
exten => 126,1,Dial(IAX2/username:password@remotehost/500)

; набираем номер, но ограничиваем звонок 5 мин
; (300 000 мс), начинаем предупреждать вызывающего
; абонента через 4 мин (240 000 мс) разговора
; и повторяем предупреждение каждые 30 с (30 000 мс)
exten => 127,1,Dial(Zap/4/2317154,,L[300000:240000:30000])
```

## Смотрите также

RetryDial()

# Dictate()

***Виртуальный диктофон***

Dictate([*базовая_папка*[,*имяфайла*]])

Это приложение обеспечивает возможность записи и воспроизведения файлов, аналогично традиционному диктофону. Параметр *базовая_папка* определяет папку, в которую Asterisk будет сохранять записываемые файлы. Если она не задана, по умолчанию файлы сохраняются в подпапку dictate папки очереди Asterisk (как задано в файле asterisk.conf).

Если задан параметр *имяфайла*, он будет использоваться при записи файла. Если он не задан, Asterisk предложит вызывающему абоненту ввести числовое имя файла.

Asterisk записывает файлы без сжатия, без заголовков, в формате signed-linear. Если потребуется преобразовать файл в другой формат, можно использовать внешнюю утилиту, такую как sox, или применить команду file convert из интерфейса командной строки Asterisk.

Приложение Dictate() имеет два основных режима: режим записи и режим воспроизведения. Для переключения режимов вызывающий абонент может нажать кнопку 1. В обоих режимах кнопка 0 используется для вызова справочной системы. Кнопка * служит для приостановки или продолжения записи либо воспроизведения. Кнопка # позволяет вызывающему абоненту выбрать новое имя файла. В режиме записи можно использовать кнопку 8, чтобы стереть всю запись и начать заново.

В режиме воспроизведения кнопка 7 обеспечивает перемотку на несколько кадров назад, а кнопка 8 – перемотку на несколько кадров вперед. Кнопка 2 используется для переключения скорости воспроизведения (1×, 2×, 3× или 4×).

```
; начинаем диктовку и сохраняем файлы в папке /tmp/dictate
exten => 123,1,Dictate(/tmp/dictate)
```

### Смотрите также

Playback(), Background(), ControlPlayback()

## Directory()

***Предоставляет справочник набираемых добавочных номеров***

Directory(*vm-контекст*[,*контекст-набора*[,*опции*]])

Предоставляет пользователям справочник добавочных номеров, которые они могут выбирать по имени абонента. Список имен и добавочных номеров можно найти в файле voicemail.conf. Аргумент *vm-контекст* является обязательным; он определяет используемый контекст voicemail.conf.

Аргумент *контекст-набора* – это контекст, используемый для выполнения звонков абонентам. Если не задан, по умолчанию принимается значение *vm-контекст*. Если для аргумента *опции* задано значение f, Asterisk будет искать соответствующий номер в справочнике на основании имени, указанного в файле voicemail.conf, а не фамилии абонента. Если задана опция e, кроме имени абонента, Asterisk будет выполнять поиск также и по добавочному номеру.

Если пользователь вводит 0 (нуль) и в текущем контексте есть добавочный номер o (или строчная буква o), управление вызовом перейдет в этот добавочный номер. Введение * обеспечит аналогичный выход, но только в добавочный номер a, что очень похоже на поведение Voicemail().

```
exten => *,1,Directory(default,incoming)
exten => #,1,Directory(default,incoming,f)
exten => 9,1,Directory(default,incoming,fe)
```

### Смотрите также

voicemail.conf

# DISA()

***Direct Inward System Access (Прямой внутрисистемный доступ): позволяет внешним абонентам выполнять исходящие звонки***

DISA(*пароль*[,*контекст*[,*callerid*[,*почтовыйящик*[@*vmконтекст*]]]])
DISA(*файл-пароля*[,*callerid*[,*почтовыйящик*[@*vmконтекст*]]])

Позволяет внешним абонентам получать «внутренний» тональный сигнал системы и совершать вызовы с АТС так, как будто они являются ее локальными пользователями. Абоненту предоставляется тональный сигнал, после чего он должен ввести свой секретный код и нажать в конце кнопку #. Если секретный код правильный, абонент услышит тональный сигнал системы и сможет выполнять звонки.

Очевидно, что такой тип доступа создает серьезную угрозу безопасности. Его необходимо использовать *чрезвычайно* осторожно, чтобы не ухудшить безопасность своей системы телефонной связи.

Аргумент *пароль* – это числовой секретный код, который абонент должен ввести, чтобы получить возможность выполнять исходящие звонки. Используя этот синтаксис, все абоненты данного добавочного номера будут применять один пароль. Чтобы абоненты могли использовать DISA() без пароля, напишите вместо пароля строку no-password (пароль не нужен).

Аргумент *контекст* определяет контекст, в котором абонент будет набирать номер. Если контекст не задан, по умолчанию приложение DISA() использует контекст disa.

Аргумент *callerid* определяет новую строку Caller ID, которая будет использоваться для исходящего звонка.

Аргумент *почтовыйящик* – это номер почтового ящика (и необязательный контекст голосовой почты, *vmконтекст*) голосовой почты. Если в указанном ящике голосовой почты есть новые сообщения, вызывающий абонент будет слышать прерывистый тональный сигнал.

Кроме того, можно использовать альтернативный синтаксис и передавать вместо аргументов *пароль* и *контекст* имя глобального файла паролей. В каждой строке файла может содержаться или секретный код, или секретный код и контекст, разделенные символом вертикальной черты (|). Если контекст не задан, приложение по умолчанию использует контекст disa.

Если регистрация абонента прошла успешно, приложение проводит синтаксический разбор набранного номера в заданном контексте:

```
; позволяем внешним абонентам звонить на номера 1-800,
; если они знают секретный код. Задаем им Caller ID, чтобы
; создать впечатление, что они звонят из компании
[incoming]
```

```
exten => 123,1,DISA(4569,disa,"Company ABC" <(234) 123-4567>)
[disa]
exten => _1800NXXXXXX,1,Dial(Zap/4/${EXTEN})
```

### Смотрите также

`Authenticate()`, `VMAuthenticate()`

## DumpChan()

***Выводит информацию о вызывающем канале в консоль***

`DumpChan([`*минимальный_уровень_детальности*`])`

Выводит на экран информацию о вызывающем канале, а также список всех переменных канала. Если задан параметр *минимальный_уровень_детальности*, вывод отображается, только когда текущий уровень детальности сообщений такой же или выше.

Если задано несколько переменных канала, `DumpChan()` покажет только первые 1024 символа списка переменных вашего канала.

```
exten => s,1,Answer()
exten => s,2,DumpChan()
exten => s,3,Background(enter-ext-of-person)
```

### Смотрите также

`NoOp()`, `Verbose()`

## EAGI()

См. `AGI()`.

## Echo()

***Воспроизводит вызывающему абоненту то, что он говорит***

`Echo()`

Возвращает звуковой сигнал из канала назад в канал. Это приложение часто используется для тестирования задержки и качества голоса VoIP-линии. Вызывающий абонент может нажать кнопку #, чтобы выйти.

```
exten => 123,1,Echo()
exten => 123,2,Playback(vm-goodbye)
```

### Смотрите также

`Milliwatt()`

# EndWhile()

***Завершает цикл `while`***

`EndWhile()`

Возвращается к ранее вызванному приложению `While()`. Полную информацию о том, как использовать цикл `while`, можно найти в описании приложения `While()`.

```
exten => 123,1,Set(COUNT=1)
exten => 123,2,While($[ ${COUNT} < 5 ])
exten => 123,3,SayNumber(${COUNT})
exten => 123,4,Set(COUNT=$[${COUNT} + 1]
exten => 123,5,EndWhile()
```

## Смотрите также

`While()`, `ExitWhile()`, `GotoIf()`

# Exec()

***Динамически выполняет приложение Asterisk***

`Exec(`*имяприложения*`(`*аргументы*`))`

Позволяет вызывать любое приложение, даже если оно не указано в коде диалплана. Выполняет выход так же, как и базовое приложение, или аварийно, если это приложение не найдено. Значение *аргументы* передается в вызываемое приложение.

Это приложение позволяет динамически вызывать приложения, извлекая их из базы данных или другого внешнего источника.

```
exten => 123,1,Set(MYAPP=SayDigits(12345))
exten => 123,2,Exec(${MYAPP})
```

## Смотрите также

`EVAL`, `TryExec()`, `ExecIf()`

# ExecIf()

***Выполняет приложение Asterisk по условию***

`ExecIf(`*выражение*`,`*приложение*`,`*аргументы*`)`

Если выражение истинно, выполняет заданное приложение, передавая в него аргументы, и возвращает результат. Больше информации о выражениях Asterisk можно найти в главе 6 или в файле channelvariables.txt в подпапке doc/ папки исходного кода Asterisk.

Если выражение ложно, выполнение продолжается со следующего приоритета.

```
exten => 123,1,ExecIf($[ ${CALLERIDNUM} = 101 ],SayDigits,12345)
exten => 123,2,SayDigits(6789)
```

### Смотрите также

EVAL, Exec(), TryExec()

## ExitWhile()

***Выполняет выход из цикла While() независимо от того, было удовлетворено условие или нет***

ExitWhile()

Завершит цикл While() независимо от того, было ли удовлетворено условие в выражении.

```
exten => 123,1,Set(COUNT=1)
exten => 123,n,While($[${COUNT} < 5])
exten => 123,n,GotoIf($[${COUNT} != 3]?continue)
exten => 123,n,ExitWhile()
exten => 123,n(continue),NoOp()
exten => 123,n,SayNumber(${COUNT})
exten => 123,n,Set(COUNT=$[${COUNT} + 1])
exten => 123,n,EndWhile()
```

### Смотрите также

While(), ContinueWhile(), EndWhile()

## ExtenSpy()

***Прослушивает аудиосигнал по добавочному номеру и может (опционально) посылать сигнал в вызывающий канал***

ExtenSpy([*добавочныйномер@контекст*[,*опции*]])

Это приложение используется для прослушивания аудиосигнала, поступающего в канал и из канала Asterisk. Для прослушивания будут выбираться только каналы, созданные исходящими вызовами с заданного добавочного номера.

При прослушивании канала могут выполняться следующие действия:

- Набор # циклически меняет уровень громкости.
- Набор * заставит приложение перейти к прослушиванию следующего доступного канала.

Параметр *опции* может содержать нуль или более следующих опций:

b

Прослушивать только соединенные каналы.

g(*группа*)

Прослушивать только каналы с переменной канала SPYGROUP, в которой в необязательном списке с разделяющими двоеточиями должна быть указана группа.

q

Скрытый режим. Указывает приложению, начиная прослушивание, не подавать звуковой сигнал или не читать имя выбранного канала.

r[(*базовоеимя*)]

Записывает разговор, ведущийся по каналу, в папку очереди для записей разговоров (обычно это /var/spool/asterisk/monitor). Необязательный аргумент *базовоеимя* задает базовое имя файла для записей, значение по умолчанию – chanspy.

v([*значение*])

Настраивает громкость прослушиваемого аудиосигнала. Значение должно быть в диапазоне от 4 до –4. Отрицательное значение сделает звук тише, тогда как положительное – громче.

w

Режим шепота. Позволяет прослушивающему каналу «говорить» с прослушиваемым каналом. При этом ни один другой соединенный канал не сможет слышать этот разговор.

W

Закрытый режим шепота. Позволяет прослушивающему каналу «говорить» с прослушиваемым каналом без возможности слышать аудиосигнал из прослушиваемого канала.

```
; Прослушиваем каналы, созданные добавочным номером 125
; в контексте lab
    exten => 123,1,ExtenSpy(125@lab,w)
```

### Смотрите также

ChanSpy()

## ExternalIVR()

***Обеспечивает сопряжение с внешним IVR-приложением***

ExternalIVR(*команда*[,*аргумент1*[,*аргумент2...*]])

Создает процесс для выполнения указанной совместимой с интерфейсом ExternalIVR команды и запускает генератор для канала. Список воспроизведения генератора управляет внешним приложением, которое может добавлять и удалять записи посредством простых команд, передаваемых по STDOUT. Внешнее приложение будет получать уведомления обо всех DTMF-событиях, полученных по каналу, а также уведомление о том, что абонент повесил трубку. Приложение не будет принудительно завершаться, когда абонент повесил трубку.

Спецификацию интерфейса ExternalIVR можно найти в папке исходного кода Asterisk в файле doc/externalivr.txt.

```
; Выполняем внешнюю программу IVR, передавая аргумент
    exten => 123,1,ExternalIVR(test_program,${MYARGUMENT})
```

## Смотрите также

AGI()

# FastAGI()

***Удаленно выполняет совместимый с AGI сценарий***

FastAGI(agi://*имяхоста*[:*порт*][/*сценарий*],*аргументы*)

Выполняет совместимую с AGI программу по сети. Это приложение очень похоже на AGI(), за исключением того что вызывает специально написанный сценарий FastAGI по сетевому соединению. Основные цели использования FastAGI – перенести требующие интенсивной работы ЦП AGI-сценарии на удаленные серверы и сократить время запуска AGI-сценария (программа FastAGI выполняется уже до того, как Asterisk соединится с ней).

FastAGI() пытается подключиться прямо к выполняющейся программе FastAGI, которая должна слушать соединения по заданному порту сервера, заданного параметром *имяхоста*. Если порт не задан, по умолчанию используется порт 4573. Если сценарий задан, он передается в программу FastAGI как переменная agi_network_script (сетевой agi-сценарий). Параметр *аргументы* будет передан в программу.

Пример сценария FastAGI можно найти в папке исходного кода Asterisk agi/fastagi-test. Он должен послужить вам хорошим образцом для написания собственных программ FastAGI.

Возвращает –1, если приложение запросило разорвать соединение, или 0 при выходе без разрыва соединения.

```
; соединяемся с программой fastagi-test, которая уже
; должна выполняться на локальном компьютере
exten => 123,1,Answer()
exten => 123,2,FastAGI(agi://localhost)

; соединяемся со сценарием FastAGI на хосте calvin
; через порт 8000 и передаем имя сценария testing
; с аргументом 12345
exten => 124,1,Answer()
exten => 124,2,FastAGI(agi://calvin:8000/testing,12345)
```

## Смотрите также

AGI(), DeadAGI()

## Festival()

***Использует механизм речевого воспроизведения текста Festival для чтения текста вызывающему абоненту***

`Festival(`*текст*`[,` *кнопкипрерывания*`])`

Подключается к выполняемому локально серверу Festival, посылает ему текст и воспроизводит результирующий звуковой файл абоненту. Это приложение позволяет вызывающему абоненту нажимать определенную кнопку (заданную параметром *кнопкипрерывания*), чтобы немедленно прекратить воспроизведение и возвратить значение *кнопкипрерывания*. Если для параметра *кнопкипрерывания* задано значение `any` (любой), `Festival()` передаст управление каналом добавочному номеру, введенному пользователем.

Более подробную информацию об использовании Festival с Asterisk можно найти в главе 14 и в файле README.festival, находящемся в подпапке contrib/ папки исходного кода Asterisk.

Сервер Festival должен быть запущен до запуска Asterisk, и, прежде чем вызывать `Festival()`, необходимо ответить каналу, используя приложение `Answer()`.

```
exten => 123,1,Answer()
exten => 123,2,Festival('This is sample speech from Festival',#)
```

## Flash()

***Посылает мгновенное событие сброса в магистральный канал Zap***

`Flash()`

Посылает мгновенное событие сброса в Zap-канал. Это инструмент для тех, кто хочет перадресовывать вызовы и выполнять другие действия, которые требуют мгновенного сброса, через AGI-сценарий. В целом это довольно бесполезное приложение.

В случае успешного выполнения возвращает 0 или –1, если это не магистральный канал Zap.

```
exten => 123,1,Flash()
```

## FollowMe()

***Функциональность «найди меня/следуй за мной»***

`FollowMe(`*followmeid*`[,` *опции*`])`

Это приложение пытается определить местоположение вызываемого абонента, набирая множество разных номеров последовательно или одновременно, как определено в файле followme.conf.

Параметр *followmeid* идентифицирует раздел файла followme.conf, в котором определено, как должен быть найден этот вызываемый абонент. Параметр *опции* может содержать нуль или более следующих значений:

s

Перед выполнением шагов функции «следуй за мной» воспроизводится сообщение о состоянии на входе.

a

Записывается имя вызывающего абонента, чтобы оно могло быть объявлено вызываемому абоненту на каждом шаге.

n

Воспроизводится сообщение о состоянии «недоступен», если все шаги для установления связи с вызываемым абонентом исчерпаны или вызываемый абонент желает быть недоступным:

```
exten => 123,1,Answer()
exten => 123,2,FollowMe(123,san)
exten => 123,3,VoiceMail(123,u)
```

## ForkCDR()

***Создает дополнительную запись CDR из текущего вызова***

`ForkCDR([опции])`

Создает дополнительную запись параметров вызова для оставшейся части текущего вызова.

Это приложение часто используется в приложениях телефонных карточек, чтобы отличить входящий вызов (исходная CDR) от оплачиваемого разговора (вторая CDR).

Если задана опция `v`, все переменные CDR из текущей записи будут унаследованы новой записью CDR.

```
exten => 123,1,Answer()
exten => 123,2,ForkCDR(v)
exten => 123,3,Playback(tt-monkeys)
exten => 123,4,Hangup()
```

### Смотрите также

Функция CDR, `NoCDR()`, `ResetCDR()`

## GetCPEID()

***Получает CPE ID от телефона, поддерживающего ADSI***

`GetCPEID()`

Получает CPE ID и другую информацию и отображает ее в консоли Asterisk. Эта информация часто нужна, чтобы правильно настроить в файле zapata.conf операции при неподнятой трубке для телефонов, поддерживающих ADSI.

Возвращает –1 только в случае разрыва связи.

```
; используем этот добавочный номер, чтобы получить
; необходимую информацию для настройки ADSI-телефонов
exten => 123,1,GetCPEID()
```

### Смотрите также

ADSIProg(), adsi.conf, zapata.conf

## Gosub()

***Переходит в новую точку, сохраняя адрес возврата***

Gosub(*контекст*,*добавочныйномер*,*приоритет*)
Gosub(*добавочныйномер*,*приоритет*)
Gosub(*приоритет*)

Переходит в заданную точку, аналогично Goto(), за исключением того что Gosub() сохраняет адрес возврата, чтобы вернуться в него позже посредством вызова Return().

### Смотрите также

GosubIf(), Macro(), Goto(), Return(), StackPop()

## GosubIf()

***Переходит в новую точку по условию, сохраняя адрес возврата***

GosubIf(*условие*?*меткаеслиистинно*:*меткаеслиложно*)

На основании вычисленного условия Gosub будет передавать выполнение или в *меткаеслиистинно*, или в *меткаеслиложно*. Вернуться в эту точку диалплана можно, вызвав позднее Return.

Слово метка (label) часто используется, чтобы обозначить возможность задать *приоритет*; *добавочный номер* и *приоритет* или *контекст*, *добавочный номер* и *приоритет*. Мы используем слово метка, чтобы не повторять каждый раз все возможные варианты.

```
; Задаем исходящий Caller*ID по умолчанию,
; если он не задан конкретным каналом.
exten => _NXXXXXX,1,GosubIf($["${CALLERID(num)}" = ""]?setcallerid,1)
exten => _NXXXXXX,n,Dial(Zap/g1/${EXTEN})
exten => _1NXXNXXXXXX,1,GosubIf($[„${CALLERID(num)}" = „"]?setcallerid,1)
exten => _1NXXNXXXXXX,n,Dial(Zap/g1/${EXTEN})
exten => setcallerid,1,Set(CALLERID(num)=6152345678)
exten => setcallerid,n,Return
```

### Смотрите также

Gosub(), Return(), MacroIf(), IF, GotoIf(),

# Goto()

***Направляет вызов в заданный приоритет, добавочный номер и контекст***

Goto([[*контекст*,]*добавочныйномер*,]*приоритет*)
Goto(*именованный_приоритет*)

Передает управление текущим каналом в заданный приоритет, при этом могут быть заданы (необязательно) вызываемый *добавочныйномер* и *контекст*.

Можно (необязательно) использовать приложение для перехода в именованный приоритет, заданный аргументом *именованный_приоритет*. Применение именованных приоритетов возможно только в рамках текущего добавочного номера.

```
exten => 123,1,Answer()
exten => 123,2,Set(COUNT=1)
exten => 123,3,SayNumber(${COUNT})
exten => 123,4,Set(COUNT=$[ ${COUNT} + 1 ])
exten => 123,5,Goto(3)

; то же самое, но с использованием именованного приоритета
exten => 124,1,Answer()
exten => 124,2,Set(COUNT=1)
exten => 124,3(repeat),SayNumber(${COUNT})
exten => 124,4,Set(COUNT=$[ ${COUNT} + 1 ])
exten => 124,5,Goto(repeat)
```

## Смотрите также

GotoIf(), GotoIfTime(), Gosub(), Macro()

# GotoIf()

***Выполняет переход по условию в заданный приоритет***

GotoIf(*условие*?*метка1*:*метка2*)

Направляет вызов в *метка1*, если условие истинно, или в *метка2*, если условие ложно. Параметры *метка1* и *метка2* могут быть опущены (в этом случае просто не выполняется переход при том или ином условии), но не оба одновременно.

В качестве метки может использоваться одно из нижеперечисленного:

- Приоритет, например 10.
- Добавочный номер и приоритет, например 123,10.
- Контекст, добавочный номер и приоритет, например incoming, 123,10.
- Именованный приоритет в рамках того же добавочного номера, например passed.

Все типы меток описаны в данном примере:

```
[globals]
; присвоим TEST какое-то значение, кроме 101, чтобы
; увидеть, что делает GotoIf(), когда условие ложно
TEST=101
;
[incoming]
; задаем переменную
; переходим в приоритет 10, если ${TEST} равна 101,
; в противном случае переходим в приоритет 20
exten => 123,1,GotoIf($[ ${TEST} = 101 ]?10:20)
exten => 123,10,Playback(the-monkeys-twice)
exten => 123,20,Playback(tt-somethingwrong)
;
; то же самое, что было сделано выше, но на этот раз
; зададим добавочный номер и приоритет для каждой метки
exten => 124,1,GotoIf($[ ${TEST} = 101 ]?123,10:123,20)
;
; то же самое, что было сделано выше, но эти метки имеют
; контекст, добавочный номер и приоритет
exten => 125,1,GotoIf($[ ${TEST} = 101 ]?incoming,123,10:incoming,123,20)
;
; то же самое, что было сделано выше, но на этот раз
; будем переходить в именованные приоритеты
exten => 126,1,GotoIf($[ ${TEST} = 101 ]?passed:failed)
exten => 126,15(passed),Playback(the-monkeys-twice)
exten => 126,25(failed),Playback(the-monkeys-twice)
```

### Смотрите также

Goto(), GotoIfTime(), GosubIf(), MacroIf()

## GotoIfTime()

***Выполняет переход по условию на основании времени и дня***

GotoIfTime(*время*,*дни_недели*,*дни_месяца*,*месяцы*?*метка*)

Выполняет переход в заданный добавочный номер, если текущее время соответствует заданному. Каждый элемент может быть определен или как * (для всех случаев), или как диапазон.

Аргументы приложения:

*время*

Диапазоны времени в 24-часовом формате.

*дни_недели*

Дни недели (mon, tue, wed, thu, fri, sat, sun).

*дни_месяца*

Дни месяца (1-31).

*месяцы*

Месяцы (jan, feb, mar, apr и т. д.).

```
; Если мы открыты, переходим в контекст open
; Мы открыты с 9 утра до 6 вечера
; с понедельника по пятницу
exten => s,1,GotoIfTime(09:00-17:59,mon-fri,*,*?open,s,1)
;
; Также мы задерживаемся по вторникам и четвергам
exten => s,n,GotoIfTime(09:00-19:59,tue&thru,*,*?open,s,1)
;
; Также мы открыты с 9 утра до полудня по субботам
exten => s,n,GotoIfTime(09:00-11:59,sat,*,*?open,s,1)
;
; Во все остальное время мы закрыты
exten => s,n,Goto(closed,s,1)
```

### Смотрите также

`GotoIf()`, `IFTIME`

## Hangup()

***Безусловный разрыв связи по текущему каналу***

`Hangup(`*`код-причина`*`)`

Безусловно разъединяет текущий канал. Для удаленного конца соединения будет задан параметр *код-причина* как причина завершения вызова, если он поддерживается каналом. По умолчанию параметр *код-причина* принимает значение `16` (нормальное завершение вызова). Допустимые значения параметра *код-причина*:

`16`

Нормальное завершение вызова.

`17`

Занято.

`19`

Нет ответа.

`21`

Вызов отклонен.

`34`

Линия перегружена.

```
exten => 123,1,Answer()
exten => 123,2,Playback(im-sorry)
exten => 123,3,Hangup()
```

### Смотрите также

`Answer()`, `Busy()`, `Congestion()`

## HasNewVoicemail()

***Проводит проверку на наличие нового сообщения голосовой почты в указанном ящике голосовой почты***

HasNewVoicemail(*vmящик*[@*контекст*][:*папка*][,*имяпеременной*[,*опции*]])

Приложение было признано устаревшим и заменено функцией VMCOUNT().

Аналогично приложению HasVoicemail(). Это приложение задает для VMSTATUS значение 1 или 0, чтобы обозначить наличие нового (непрослушанного) сообщения в ящике голосовой почты, заданном аргументом *vmящик*. Аргумент *контекст* соответствует контексту голосовой почты, а *папка* соответствует папке голосовой почты. Если папка голосовой почты не задана, используется папка по умолчанию, INBOX. Если присутствует аргумент *имяпеременной*, HasNewVoicemail() сохраняет в заданную переменную количество сообщений в заданной папке.

Если для аргумента *опции* задано значение j, Asterisk в случае наличия нового сообщения голосовой почты будет направлять вызов в приоритет n + 101.

```
; проверяем наличие непрослушанного сообщения
; голосовой почты в INBOX почтового ящика 123
; в контексте голосовой почты по умолчанию
exten => 123,1,Answer()
exten => 123,n,HasNewVoicemail(123@default)
exten => 123,n,GotoIf($[${HASVMSTATUS} > 0]?newvm)
exten => 123,n,Playback(vm-youhave)
exten => 123,n,Playback(vm-no)
exten => 123,n,Playback(vm-messages)
exten => 123,n,Goto(done)
exten => 123,n(newvm),Playback(vm-youhave)
exten => 123,n,SayNumber(${HASVMSTATUS})
exten => 123,n,Playback(vm-INBOX)
exten => 123,n,Playback(vm-messages)
exten => 123,n(done),NoOp()
```

### Смотрите также

HasVoicemail(), MailboxExists(), VMCOUNT

## HasVoicemail()

***Показывает, имеются ли сообщения голосовой почты в указанном ящике голосовой почты***

HasVoicemail(*vmящик*[@*контекст*][:*папка*][|*имяпеременной*[,*опции*]])

Задает значение переменной канала HASVMSTATUS, чтобы информировать о наличии сообщений в ящике голосовой почты *vmящик*. Аргумент *кон-*

*текст* соответствует контексту голосовой почты, а *папка* – папке голосовой почты. Если папка не задана, используется папка по умолчанию, INBOX. Если передается аргумент *имяпеременной*, приложение сохраняет в этой переменной количество сообщений в указанной папке.

Если для аргумента *опции* задано значение `j`, Asterisk в случае наличия голосовой почты в заданной папке будет направлять вызов в приоритет `n` + 101.

```
; проверяем, есть ли хотя бы одно сообщение
; голосовой почты в INBOX почтового ящика 123
; в контексте голосовой почты по умолчанию
exten => 123,1,Answer()
exten => 123,2,HasVoicemail(123@default,COUNT)
exten => 123,3,GotoIf(${VMSTATUS}?1000)
exten => 123,4,Playback(vm-youhave)
exten => 123,5,Playback(vm-no)
exten => 123,6,Playback(vm-messages)
exten => 123,1000,Playback(vm-youhave)
exten => 123,1001,SayNumber($COUNT)
exten => 123,1002,Playback(vm-messages)
```

### Смотрите также

`HasNewVoicemail()`, `MailboxExists()`

## IAX2Provision()

***Выполняет подготовку вызывающего IAXy-устройства***

`IAX2Provision([`*шаблон*`])`

Выполняет подготовку к работе вызывающего устройства IAXy (предполагая, что вызывающий субъект – это IAXy) с использованием заданного шаблона. Если шаблон не задан, используется шаблон по умолчанию. Шаблоны подготовки IAXy описаны в конфигурационном файле iaxprov.conf.

```
; выполняем подготовку IAXy-устройств с использованием
; шаблона по умолчанию, когда они набирают этот
; добавочный номер
exten => 123,1,IAX2Provision(default)
```

## ICES()

***Выполняет потоковую передачу аудиоданных на сервер Icecast***

`ICES([`*конфигурация*`])`

Выполняет потоковую передачу аудиоданных на сервер Icecast (или совместимый) с помощью приложения `ices`.

См. файл-образец config в contrib/asterisk-ices.xml.

```
exten => 123,1,Answer()
exten => 123,n,ICES(/tmp/my-ices-config.xml)
```

# ImportVar()

***Задает переменную на основании переменной канала, полученной от другого канала***

ImportVar(*новаяпеременная=канал*,*значение*)

Задает переменной *новаяпеременная* значение, вычисленное для заданного канала (не текущего канала). Если *новаяпеременная* начинается с символа _, предполагается простое наследование. Если она начинается с символов _ _, предполагается множественное наследование.

```
; читаем информацию Caller ID из канала Zap/1
exten => 123,1,Answer()
exten => 123,n,ImportVar(cidinfo=Zap/1,CALLERID(all))
```

## Смотрите также

Set()

# Log()

***Протоколирует специальное сообщение из диалплана***

Log(*уровень*|*сообщение*)

Посылает специальное сообщение из диалплана в файлы журнала. Это приложение может быть полезным для протоколирования в файлах журнала исключительных обстоятельств с целью последующего их изучения. Параметр *уровень* может принимать одно из следующих значений:

DEBUG (отладка)

Сообщение отладки. Обычно не регистрируется в системе, находящейся в производственной эксплуатации.

NOTICE (извещение)

Информационное сообщение.

WARNING (предупреждение)

Обстоятельство, которое может быть серьезным, но не является ошибкой.

ERROR (ошибка)

Что-то пошло совсем не так.

## Смотрите также

NoOp(), Verbose()

# LookupBlacklist()

***Осуществляет поиск имени/номера Caller ID в черном списке***

LookupBlacklist([*опции*])

Это приложение признано устаревшим и заменено приложением GotoIf(${BLACKLIST()}?*контекст*|*добавочныйномер*|*приоритет*).

Выполняет поиск номера Caller ID для активного канала в базе данных Asterisk (в семействе blacklist). Если номер Caller ID обнаружен в черном списке, Asterisk задает для переменной канала LOOKUPBL STATUS значение FOUND (найден). В противном случае для переменной задается значение NOTFOUND (не найден).

Если в параметре *опции* используется опция j и номер найден и если существует приоритет n + 101 (где n – текущий приоритет), канал должен будет продолжить выполнение с этого приоритета.

Чтобы пополнить черный список из CLI Asterisk, введите команду `database put blacklist` *`имя / номер`*.

```
; направляем номера черного списка в бесконечный цикл,
; в противном случае набираем номер, заданный переменной
; ${JOHN}
exten => 123,1,Answer()
exten => s,2,LookupBlacklist()
   ; если номер Caller ID обнаружен в черном списке, переходим к метке goaway
   exten => 123,n,GotoIf($["${LOOKUPBLSTATUS}" = "FOUND"]?goaway)
   ; в противном случае продолжаем и звоним Джону
exten => 123,n,Dial(${JOHN})
exten => 123,n(goaway),Busy(5)
exten => 123,n,Hangup()
```

### Смотрите также

BLACKLIST

## LookupCIDName()

***Выполняет поиск имени Caller ID в AstDB***

LookupCIDName()

Это приложение было признано устаревшим и заменено приложением Set(CALLERID(*имя*)=${DB(cidname/${CALLERID(*номер*)})}).

Использует номер Caller ID активного канала для извлечения имени Caller ID из AstDB (семейство cidname). Это приложение не делает ничего, если по каналу не получен Caller ID. Это полезно, если вы не подписаны на доставку имени Caller ID или хотите менять имена Caller ID для некоторых входящих вызовов.

```
; ищем информацию Caller ID в AstDB и передаем ее
; на телефон Джейн
exten => 123,1,Answer()
```

```
exten => 123,2,LookupCIDName()
exten => 123,3,Dial(SIP/Jane)
```

### Смотрите также

DB

## Macro()

***Вызывает заранее определенный макрос диалплана***

Macro(*имямакроса*,*arg1*,*arg2*...)

Выполняет макрос, определенный в контексте macro-*имямакроса*, переходя в добавочный номер s этого контекста и выполняя каждый шаг. Возвращается после того, как все шаги были выполнены.

Вызывающий добавочный номер, контекст и приоритет хранятся в переменных ${MACRO_EXTEN}, ${MACRO_CONTEXT} и ${MACRO_PRIORITY} соответственно. Аргументы *arg1*, *arg2* и т. д. становятся в контексте макроса переменными ${ARG1}, ${ARG2} и т. д.

Macro() завершается аварийно, если на любом из шагов макроса возник сбой или выявлен разрыв соединения. Если по завершении работы макроса задана переменная ${MACRO_OFFSET}, это приложение попытается продолжить выполнение с приоритета MACRO_OFFSET + n + 1, если таковой существует, или с n + 1 в противном случае. (В обоих случаях n означает текущий приоритет.)

Если внутри макроса вызывается приложение Goto() для перехода в контекст вне выполняющегося в настоящий момент макроса, макрос будет завершен, а управление передано в точку, указанную в параметрах Goto().

```
; определяем макрос для обратного счета
; от заданного значения
[macro-countdown]
exten => s,1,Set(COUNT=${ARG1})
exten => s,2,While($[ ${COUNT} > 0])
exten => s,3,SayNumber(${COUNT})
exten => s,4,Set(COUNT=$[ ${COUNT} - 1 ])
exten => s,5,EndWhile()

; вызываем наш макрос с двумя разными значениями
[example]
exten => 123,1,Macro(countdown,10)
exten => 124,1,Macro(countdown,5)
```

На время выполнения макрос становится текущим контекстом. Это означает, что в случае разрыва соединения, например, поиск добавочного номера h будет выполняться в макросе, а не в контексте, из которого этот макрос был вызван. Поэтому в макросе должны быть обязательно описаны все соответствующие добавочные номера (в AEL можно использовать catch).

Из-за способа реализации Macro() (оно выполняет содержащиеся в нем приоритеты через вспомогательный механизм) и выделения фиксированного стека памяти на каждый поток макрос ограничен семью уровнями вложенности (макрос, вызывающий макрос, вызывающий макрос и т. д.). Приложения, интенсивно использующие стек, в глубоко вложенном макросе могут привести к сбою в работе Asterisk раньше достижения этого предельного уровня вложенности.

### Смотрите также

MacroExit(), Goto(), Gosub(), главу 6

## MacroExclusive()

***Выполняет макрос исключительно для одного канала***

MacroExclusive(*имямакроса*[,*аргументы*])

Выполняет заданный макрос, гарантируя, что одновременно этот макрос выполняется только одним каналом. Если другой канал уже выполняет этот макрос, MacroExclusive() приостановит выполнение данного канала до тех пор, пока тот канал не выйдет из макроса.

### Смотрите также

Macro()

## MacroExit()

***Явный выход из макроса***

MacroExit()

Выполняет явный выход из макроса. Обычно Macro() автоматически завершается, выполнив все приоритеты. MacroExit() обеспечивает возможность завершить макрос раньше.

### Смотрите также

Macro()

## MacroIf()

***Вызывает заранее определенный макрос по условию***

MacroIf(*условие*?*макросеслиистинно*,*аргументы*:*макросеслиложно*,*аргументы*)

Вычисляет *условие*, а затем выполняет *макросеслиистинно* или *макросеслиложно*. Во всем остальном, кроме вычисления условия, MacroIf() ведет себя идентично Macro().

```
; определяем макрос для обратного счета
; от заданного значения
[macro-countdown]
```

```
exten => s,1,Set(COUNT=${ARG1})
exten => s,2,While($[ ${COUNT} > 0])
exten => s,3,SayNumber(${COUNT})
exten => s,4,Set(COUNT=$[ ${COUNT} - 1])
exten => s,5,EndWhile()

; определяем макрос для прямого счета
; от заданного значения
[macro-countup]
exten => s,1,Set(COUNT=1)
exten => s,2,While($[ ${COUNT} < ${ARG1}])
exten => s,3,SayNumber(${COUNT})
exten => s,4,Set(COUNT=$[ ${COUNT} + 1])
exten => s,5,EndWhile()

; вызываем наш макрос с двумя разными значениями
[example]
exten => 123,1,MacroIf($[ ${foo} < 5 ]?countup,${foo}:countdown,${foo})
```

## Смотрите также

`GotoIf()`, `GosubIf()`, `IF`, глава 6

# MailboxExists()

***Выполняет переходы по условию, если заданный ящик голосовой почты существует***

`MailboxExists(`*`почтовыйящик`*`[@`*`контекст`*`[,`*`опции`*`]])`

Проверяет, существует ли в системе голосовой почты Asterisk почтовый ящик, определенный аргументом *`почтовыйящик`*. Если почтовый ящик находится не в контексте голосовой почты `default`, можно передать *`контекст`* для голосовой почты.

Это приложение задает переменную канала `VMBOXEXISTSSTATUS`. Если почтовый ящик существует, ей будет задано значение `SUCCESS` (успех). В противном случае она получит значение `FAILED` (неудача).

Если в качестве параметра *`опции`* передается опция `j`, в случае существования почтового ящика, указанного аргументом *`почтовыйящик`*, приложение перейдет к приоритету `n` + 101 (где `n` – текущий приоритет).

```
exten => 123,1,Answer()
exten => 123,n,Set(MYMAILBOX=123@default)
exten => 123,n,MailboxExists(${MYMAILBOX})
exten => 123,n,GotoIf($["${VMBOXEXISTSSTATUS}" = "SUCCESS"]?exists)
exten => 123,n,Playback(im-sorry)
exten => 123,n,Hangup()
exten => 123,n(exists),Voicemail(u123)
```

## Смотрите также

`HasVoicemail()`, `HasNewVoicemail()`

## MeetMe()

***Добавляет звонящего в конференцию MeetMe***

MeetMe([*номерконференции*[,*опции*[,*ПИН*]]])

Добавляет звонящего в сеанс речевой конференц-связи, обозначенный аргументом *номерконференции*. Если номер конференции опущен, пользователю будет предложено его ввести.

Если передается аргумент *ПИН*, звонящий для входа в конференцию должен ввести ПИН-код.

Строка *опции* может содержать нуль или более символов из следующего списка:

a

Задается режим администрирования.

A

Задается маркированный режим.

b

Выполняется AGI-сценарий, заданный в переменной ${MEETME_AGI_BACKGROUND}; по умолчанию это conf-background.agi. (Примечание: это не работает для не-Zap-каналов в той же конференции.)

c

При входе в конференцию нового пользователя анонсируется количество пользователей.

d

Динамически добавляет пользователя в конференцию.

D

Динамически добавляет пользователя в конференцию, предлагая ввести ПИН-код.

e

Выбирается пустая конференция.

E

Выбирается пустая конференция, для входа в которую не требуется вводить ПИН-код.

F

DTMF-коды передаются через конференцию другим участникам. DTMF-коды, используемые для активации функций конференции, передаваться не будут.

i

Анонсируется вход/выход пользователя с суммарной информацией.

I

Анонсируется вход/выход пользователя без суммарной информации.

l

Устанавливается режим «только для прослушивания» (можно только слушать, не говорить).

m

Участник вводится в конференцию с изначально выключенным микрофоном.

M

Если в конференции всего один участник, активируется музыка во время ожидания.

o

Включается оптимизация разговора. При этом Asterisk полагает, что у участников, которые не говорят в данный момент времени, отключен микрофон, то есть при передаче данных не выполняется кодировка и поступающий сигнал, не регистрируемый как речь, опускается без увеличения фонового шума.

p

Пользователь может выйти из конференции, нажав кнопку #.

P

Всегда предлагается ввести ПИН-код, даже если он задан.

q

Задается скрытый режим. В скрытом режиме Asterisk не подает звуковых сигналов при входе или выходе участников конференции.

r

Конференция записывается (как `${MEETME_RECORDINGFILE}`, используя формат `${MEETME_RECORDINGFORMAT}`). Имя файла по умолчанию – meetme-conf-rec-`${CONFNO}`-`{UNIQUEID}`, а формат по умолчанию – .wav.

s

При получении * предоставляется меню (меню пользователя или администратора в зависимости от того, отмечен ли пользователь как администратор).

t

Устанавливается режим «только для разговора» (можно только говорить, но не слушать).

T

Устанавливается выявление говорящего. Asterisk будет передавать в интерфейс Manager события, идентифицирующие канал говоря-

щего. Говорящий также будет обозначен в выводе CLI-команды `meetme list`.

`w[(`*количествосекунд*`)]`

Ожидает входа в конференцию администратора. Если параметр *количествосекунд* не задан, конференция будет ожидать входа администратора неопределенно долго. Если параметр *количествосекунд* задан, конференция ожидает в течение заданного времени. Если по истечении этого срока администратор так и не выполнил вход, обработка вызова продолжится со следующего приоритета диалплана.

`x`

Конференция закрывается, когда ее покидает последний маркированный пользователь.

`X`

Позволяет пользователю выходить из конференции, вводя действительный одноразрядный добавочный номер (задается посредством переменной `${MEETME_EXIT_CONTEXT}`) или добавочный номер в текущем контексте, если эта переменная не определена.

`1`

Не воспроизводить начальное сообщение при входе в конференцию первого участника.

```
exten => 123,1,Answer()
; добавляем звонящего в конференцию номер 501 с ПИН-кодом 1234
exten => 123,2,MeetMe(501,DpM,1234)
```

Для обеспечения работы конференц-связи MeetMe должен быть установлен подходящий интерфейс синхронизации Zaptel.

### Смотрите также

`MeetMeAdmin()`, `MeetMeCount()`

## MeetMeAdmin()

***Осуществляет администрирование конференции MeetMe***

`MeetMeAdmin(`*номерконференции*`,` *команда*`[,` *пользователь*`])`

Выполняет указанную команду администрирования MeetMe для заданной конференции. Для некоторых команд можно задать пользователя, для которого должна быть выполнена указанная команда. Команда может принимать одно следующих значений:

`e`

Исключить пользователя, присоединившегося к конференции последним.

k

Исключить из конференции указанного пользователя.

K

Исключить из конференции всех пользователей.

l

Снять блокировку конференции.

L

Блокировать конференцию.

m

Включить микрофон заданного пользователя.

M

Выключить микрофон заданного пользователя.

n

Включить микрофоны всех участников конференции.

N

Выключить микрофоны всех участников конференции, не являющихся администраторами.

r

Сбросить все настройки громкости для заданного пользователя.

R

Сбросить все настройки громкости для всех участников.

s

Уменьшить громкость разговора для всей конференции.

S

Увеличить громкость разговора для всей конференции.

t

Уменьшить громкость разговора для заданного пользователя.

T

Увеличить громкость разговора для заданного пользователя.

u

Уменьшить громкость прослушивания для заданного пользователя.

U

Увеличить громкость прослушивания для заданного пользователя.

v

Уменьшить громкость прослушивания для всей конференции.

V

Увеличить громкость прослушивания для всей конференции.

```
; выключить звук конференции 501
exten => 123,1,MeetMeAdmin(501,N)

; исключить пользователя 1234 из конференции 501
exten => 124,1,MeetMeAdmin(501,k,1234)
```

Список участников конференции можно получить с помощью CLI-команды Asterisk `meetme list` или используя интерфейс Asterisk Manager.

### Смотрите также

`MeetMe()`, `MeetMeCount()`

## MeetMeCount()

***Подсчитывает количество пользователей, принимающих участие в конференции MeetMe***

`MeetMeCount(`*номерконференции*`[,`*переменная*`])`

Воспроизводит количество пользователей, принимающих участие в конференции MeetMe, заданной аргументом *номерконференции*. Если задана *переменная*, значение будет сохранено в эту переменную без воспроизведения.

```
; подсчитываем количество пользователей, принимающих
; участие в конференции 501, и присваиваем это число
; переменной ${COUNT}
exten => 123,1,MeetMeCount(501,COUNT)
```

### Смотрите также

`MeetMe()`, `MeetMeAdmin()`

## Milliwatt()

***Генерирует тональный сигнал частотой 1000 Гц***

`Milliwatt()`

Это приложение генерирует постоянный тональный сигнал частотой 1000 Гц с уровнем 0 дБм (μlaw). Это приложение часто используется для тестирования звуковых характеристик конкретного канала.

```
; генерируем тональный сигнал частотой 1000Гц
exten => 123,1,Milliwatt()
```

Пожалуйста, обратите внимание, что существует сервис, используемый поставщиками услуг связи для тестирования потерь в линиях и получивший в отрасли название «1000 циклов». Суть в том, что частота тонального сигнала, передаваемого оборудованием поставщика услуг, на самом деле составляет 1004 Гц, поэтому, если необходимо протестировать потери в аналоговой линии из Asterisk, `Milliwatt()` может не обеспечить точных результатов.

### Смотрите также

`Echo()`, `Playtones()`

## MixMonitor()

***Записывает разговор по каналу в фоновом режиме, синхронно объединяя оба направления разговора***

`MixMonitor(`*имяфайла.ext*`,` *опции*`,` *команда*`)`

Записывает аудиосигнал в текущем канале в заданный файл. Если в качестве аргумента *имяфайла* задан полный путь, `MixMonitor()` использует этот путь; в противном случае создает файл в заданной в asterisk.conf папке для записи разговоров.

Если задана, команда будет выполнена, когда запись будет завершена из-за разрыва соединения или в результате вызова `StopMixMonitor()`.

Параметр *опции* может содержать нуль или более следующих опций:

`a`

Разговор дописывается в конец файла, перезаписи файла при этом не происходит.

`b`

Разговор сохраняется, только если канал соединен.

Сюда не относятся конференции или звуковые файлы, воспроизводимые для каждой соединенной стороны.

`v(`*x*`)`

Громкость прослушивания меняется в *x* раз (диапазон от −4 до 4).

`V(`*x*`)`

Громкость речи меняется в *x* раз (диапазон от −4 до 4).

`W(`*x*`)`

Громкость прослушивания и речи меняется в *x* раз (диапазон от −4 до 4).

```
; Записываем разговор по каналу
exten => 123,1,MixMonitor(/var/lib/asterisk/sounds/123.wav)
```

## Смотрите также

Monitor(), StopMixMonitor(), PauseMonitor(), UnpauseMonitor()

# Monitor()

***Отслеживает (записывает) разговор по текущему каналу***

Monitor([*формат_файла*[:*базовыйurl*][,*базовое_имяфайла*][,*опции*]])

Начинает запись разговора по каналу. Входные и выходные речевые пакеты канала записываются в файлы до тех пор, пока не будет разорвана связь по каналу или запись не будет остановлена приложением StopMonitor().

Monitor() принимает следующие аргументы:

*формат_файла*

Определяет формат файла. Если не задан, по умолчанию используется wav.

*базовое_имяфайла*

Если задан, меняет используемое имя файла на заданное.

*опции*

Может быть задана одна опция из двух:

m

Когда запись закончится, выполнить слияние двух файлов, содержащих список приоритетов, в один и удалить их. Если задана переменная ${MONITOR_EXEC}, вместо soxmix будет выполнено указанное в ней приложение, и исходные файлы *не* будут удалены автоматически. soxmix (или ${MONITOR_EXEC}) принимает три аргумента: два файла приоритетов и имя создаваемого объединенного файла, которое аналогично именам исходных файлов, но без указателей in/out. Если задана переменная ${MONITOR_EXEC_ARGS}, в качестве дополнительных аргументов в ${MONITOR_EXEC} будет передано содержимое. И ${MONITOR_EXEC}, и флаг m можно задать из интерфейса администратора.

b

Не начинать запись, пока канал не будет соединен с другим каналом.

```
exten => 123,1,Answer()
; записываем разговор по текущему каналу
; и объединяем звуковые каналы в конце записи
exten => 123,2,Monitor(wav,monitor_test,mb)
exten => 123,3,SayDigits(12345678901234567890)
exten => 123,4,StopMonitor()
```

### Смотрите также

ChangeMonitor(), StopMonitor(), MixMonitor(), PauseMonitor(), UnpauseMonitor()

## MorseCode()

***Воспроизводит код Морзе***

MorseCode(*строка*)

Воспроизводит строку, записанную в виде международного кода Морзе. Перечисленные далее переменные канала будут оказывать влияние на воспроизведение:

MORSEDITLEN

Длина ТОЧКИ в миллисекундах. Значение по умолчанию – 80 мс.

Длительность всех остальных тональных сигналов и пауз определена в международном стандарте кода Морзе относительно длительности ТОЧКИ, и поэтому все остальные длительности будут настраиваться соответствующим образом.

MORSETONE

Частота, в герцах (Гц), которая будет использоваться. Значение по умолчанию – 800 Гц.

```
; тире-точка-тире точка-точка точка-точка-точка-точка-тире
; тире-точка-тире точка-точка-тире точка-тире
exten => 123,1,Answer()
exten => 123,2,MorseCode(KI4KUA)
```

### Смотрите также

SayAlpha(), SayPhonetic()

## MP3Player()

***Воспроизводит MP3-файл или поток***

MP3Player(*местоположение*)

Использует программу mpg123 для воспроизведения вызывающему абоненту файла из заданного местоположения. В качестве местоположения может быть задано или имя файла, или действительный URL. Вызывающий абонент может прекратить воспроизведение, нажав любую кнопку.

Чтобы это приложение работало правильно, должна быть установлена соответствующая версия mpg123. В настоящее время Asterisk лучше всего работает с mpg123-0.59r. Другие версии могут обеспечить результаты, не вполне соответствующие тому, что требуется.

```
exten => 123,1,Answer()
exten => 123,2,MP3Player(test.mp3)

exten => 123,1,Answer()
exten => 123,2,MP3Player(http://example.com/test.mp3)
```

## MusicOnHold()

***Неопределенно долго воспроизводит музыку во время ожидания***

`MusicOnHold(`*`класс`*`)`

Воспроизводит музыку во время ожидания, заданную аргументом *класс*, соответственно настройкам файла musiconhold.conf. Если аргумент опущен, будет использоваться музыка во время ожидания по умолчанию. Задать класс музыки по умолчанию для канала можно с помощью функции диалплана `MUSICCLASS`.

```
; перенаправляем вызовы систем продаж по телефону на этот
; добавочный номер, чтобы занять их
exten => 123,1,Answer()
exten => 123,n,Playback(tt-allbusy)
exten => 123,n,MusicOnHold(default)
```

### Смотрите также

`SetMusicOnHold()`, `WaitMusicOnHold()`, `MUSICCLASS`

## NBScat()

***Воспроизводит локальный NBS-поток***

`NBScat()`

Использует программу `nbscat8k` для прослушивания локального потока NBS (Network Broadcast Sound – широковещательная сетевая передача звука). (Более подробную информацию можно найти в модуле nbs сервера Subversion компании Digium.) Вызывающий абонент может прекратить воспроизведение, нажав любую кнопку.

При разрыве соединения возвращает –1.

```
exten => 123,1,Answer()
exten => 123,2,NBScat()
```

## NoCDR()

***Отключает запись параметров вызовов для текущего звонка***

`NoCDR()`

Отключает запись CDR для текущего звонка.

```
; не протоколировать звонки на 555-1212
exten => 5551212,1,Answer()
exten => 5551212,2,NoCDR()
exten => 5551212,3,Dial(Zap/4/5551212)
```

## Смотрите также

AppendCDRUserField(), ForkCDR(), SetCDRUserField()

# NoOp()

***Не производит никаких действий***

NoOp(*текст*)

Ничего не делает, это просто заполнитель. Данное приложение часто используется как инструмент отладки. Если уровень детальности сообщений ядра Asterisk равен 3 или более, Asterisk обрабатывает и выводит каждую строку диалплана перед ее выполнением. Это означает, что любые аргументы, переданные в приложение NoOp() (в параметре *текст*), выводятся в консоли. Опытный администратор Asterisk может использовать вывод консоли для отладки диалплана.

```
exten => 123,1,NoOp(CallerID is ${CALLERID})
```

Нет необходимости заключать текст в кавычки. Если кавычки помещены в круглые скобки, они будут выведены в консоли.

### Когда использовать NoOp() и Verbose()

Разница между Verbose() и NoOp() невелика. Приведем здесь некоторые соображения по поводу того, как разобраться, в какой ситуации следует использовать эти приложения. Приложение Verbose() пригодится, если надо что-то вывести в консоль. Используя команду set verbose (с указанием требуемого уровня детальности, от 0 до 4), можно настроить вывод так, чтобы на экране отображались не все операции системы, а лишь те, которые имеют такой же или меньший уровень детальности. (На самом деле можно задать любую детальность. Команда set verbose 999 будет прекрасно работать, но мы не нашли вывода с уровнем детальности выше, чем 4, поэтому на данный момент задавать детальность больше 4 просто не имеет смысла.) Это означает, что можно выводить всевозможную информацию, имеющую отношение к тестируемому разделу кода, без необходимости просматривать другие сообщения системы. Если записать в диалплане следующее:

```
exten => _X.,n,Verbose(2, ${SOME_VAR})
```

можно будет в CLI задать детальность 2 или меньше (core set verbose 2) и просматривать вывод различных вызовов Verbose(), но очень мало другой информации.

Подробнее об использовании Verbose() можно будет прочитать в данном приложении ниже, в посвященном ему разделе. При-

ложение NoOp() лучше всего использовать как заполнитель. Например, если в диалплане имеется Goto() с использованием метки приоритета, NoOp() можно применять как точку перехода из этого приложения. Например,

```
exten => _X.,n(call_forward),NoOp()
```

является превосходным маркером для указания перехода в диалплане в определенную точку. Из этой точки можно продолжать выполнение любой логики, которую требуется применить к этой части добавочного номера (судя по метке, речь идет о пересылке вызова). NoOp() применяется, когда вы не знаете, какие действия придется выполнять за этой меткой, и чтобы гарантированно не менять в коде саму метку. Оно никогда не будет делать ничего другого, кроме как предоставлять точку перехода для Goto(). Поэтому NoOp() можно помещать куда угодно и быть увереннным, что это не станет причиной какого-либо неожиданного поведения.

Если вы ничего не поняли, виною наша неспособность правильно описать ситуацию. Поэкспериментируйте с Verbose() и NoOp() в своем диалплане (их можно использовать где угодно) – и вы быстро разберетесь в том, как они могут помочь (особенно если вы, как и мы, допускаете много синтаксических ошибок).

## Смотрите также

Verbose(), Log()

# Page()

***Открывает одностороннюю аудиосвязь с несколькими телефонами***

Page(*технология/канал1*[&*технология/канал2*]&[...][&*технология/каналN*][|*опции*])

Размещает исходящие вызовы абонентов в каналах, заданных аргументами *технология/канал*, и вводит их в конференцию как участников с выключенными микрофонами. Исходный вызывающий абонент помещается в эту конференцию как единственный участник, который может говорить. Когда он покидает конференцию, она уничтожается. Могут быть заданы следующие опции:

d

Двусторонняя аудиосвязь. Обеспечивает возможность перечисленным в приложении Page() людям отвечать вызывающему абоненту.

q

Скрытый режим. Не воспроизводить звуковой сигнал вызывающему абоненту.

r

Записывать сообщение. Больше информации можно найти в описании опции r для команды MeetMe.

```
exten => 123,1,Page(SIP/101&SIP/102&IAX2/iaxy123)
```

### Смотрите также

MeetMe()

## Park()

***Выполняет парковку текущего вызова***

Park()

Выполняет парковку текущего вызова (обычно в сочетании с контролируемой переадресацией для определения номера парковочного слота). Это приложение всегда регистрируется системой внутренне и не требует явного введения в диалплан, хотя необходимо включить контекст parkedcalls. Настройки парковки задаются в файле features.conf.

```
; явно паркуем вызывающего абонента
include => parkedcalls
exten => 123,1,Answer()
exten => 123,n,Park()
```

### Смотрите также

ParkAndAnnounce(), ParkedCall()

## ParkAndAnnounce()

***Выполняет парковку текущего вызова и объявляет о вызове по заданному каналу***

ParkAndAnnounce(*шаблон*,*времяожидания*,*канал*[,*котекст_возврата*])

Выполняет парковку текущего вызова в парковочный слот и объявляет о вызове по заданному каналу. Шаблон – это разделенный двоеточиями список файлов, которые должны быть воспроизведены; слово PARKED (припаркован) заменяется номером слота парковки вызова. Аргумент *времяожидания* – это время в секундах, через которое вызов возвратится в *контекст_возврата*. Аргумент *канал* определяет канал, на который необходимо позвонить, чтобы сделать объявление. Console/dsp вызывает консоль. *контекст_возврата* – это метка в стиле Goto() для возвращения вызова по истечении времени ожидания, которой по умолчанию является приоритет n + 1 (где n – текущий приоритет) в контексте *контекст_возврата*.

```
include => parkedcalls
exten => 123,1,Answer()
exten => 123,2,ParkAndAnnounce(vm-youhave:a:pbx-transfer:at:vm-extension:
PARKED,120,Console/dsp)
```

```
exten => 123,3,Playback(vm-nobodyavail)
exten => 123,4,Playback(vm-goodbye)
exten => 123,5,Hangup()
```

### Смотрите также

Park(), ParkedCall()

## ParkedCall()

***Отвечает на припаркованный вызов***

ParkedCall(*парковочныйслот*)

Соединяет вызывающего абонента с припаркованным вызовом, находящимся в парковочном слоте, который обозначен аргументом *парковочныйслот*. Это приложение всегда регистрируется системой внутренне и не требует явного добавления в диалплан, хотя необходимо включить контекст parkedcalls.

```
; ответить на вызов, припаркованный
; в парковочном слоте 701
exten => 123,1,Answer()
exten => 123,2,ParkedCall(701)
```

### Смотрите также

Park(), ParkAndAnnounce()

## PauseMonitor()

***Приостанавливает запись разговора по каналу***

PauseMonitor()

Временно приостанавливает отслеживание (запись) текущего канала

```
exten => 123,1,Answer()
exten => 123,n,Monitor(wav,monitor_test)
exten => 123,n,Playback(demo-congrats)
    ; временно приостанавливаем запись, пока собираем секретную информацию
exten => 123,n,PauseMonitor()
exten => 123,n,Read(NEWPASS,vm-newpassword)
exten => 123,n,SayDigits(${NEWPASS})
exten => 123,n,UnpauseMonitor()
exten => 123,n,Dial(${JOHN})
```

### Смотрите также

Monitor(), StopMonitor(), UnpauseMonitor()

## PauseQueueMember()

***Временно блокирует участника обработки очереди входящих вызовов***

PauseQueueMember([*имяочереди*], *интерфейс*[, *опции*])

Приостанавливает выполнение заданного интерфейса обработки очереди вызовов. При этом любые вызовы из очереди не будут передаваться в этот интерфейс до тех пор, пока он не будет возвращен к работе с помощью приложения UnpauseQueueMember() или интерфейса Manager. Если аргумент *имяочереди* не задан, интерфейс блокируется во всех очередях, участником которых он является.

Это приложение по завершении выполнения задает для переменной канала PQMSTATUS значение PAUSED (приостановлен) или NOTFOUND (не найден).

Если для параметра *опции* задано значение j и данного интерфейса нет в указанной очереди или если очередь не задана и интерфейса нет ни в одной очереди, выполнение перейдет в приоритет n + 101 (где n – текущий приоритет), если таковой существует.

```
exten => 123,1,PauseQueueMember(,SIP/300)
exten => 124,1,UnpauseQueueMember(,SIP/300)
```

### Смотрите также

UnpauseQueueMember()

## Pickup()

***Отвечает на звонок с другого телефона***

Pickup(*добавочныйномер*[@*контекст*][&*добавочныйномер2*[@*контекст2*][...])

Отвечает на все звонки, поступающие на номер, указанный в аргументе *добавочныйномер*. Если указано несколько добавочных номеров, Pickup() будет извлекать первый подходящий. Если аргумент *контекст* не задан, будет использоваться текущий контекст.

Существует также специальный контекст PICKUPMARK. Если он задан, Pickup будет находить первый вызывающий канал, для которого значение переменной канала PICKUPMARK соответствует значению аргумента *добавочныйномер*.

## Playback()

***Воспроизводит заданный аудиофайл вызывающему абоненту***

Playback(*имяфайла*[&*имяфайла2*...][,*опции*])

Воспроизводит заданный посредством аргумента *имяфайла* файл вызывающему абоненту. Значение *имяфайла* не должно включать расширения файла, поскольку Asterisk автоматически выберет аудиофайл, преобразование которого пройдет с минимальными затратами. Также может быть включено нуль или более опций. Если задана опция skip (пропустить), сообщение воспроизводится только для канала, находящегося в состоянии «отвечено» (то есть если установлено соединение). При заданной опции skip приложение немедленно возвращается, если канал не соединен. В противном случае, если не задана опция noanswer, канал будет переведен в состояние «отвечено» и после этого будет вос-

произведен аудиофайл. (Не все каналы поддерживают воспроизведение сообщений, когда соединение еще не установлено.) Если в качестве одной из опций передана j и указанного файла не существует, это приложение переходит в приоритет n + 101 (где n – текущий приоритет), если таковой существует.

```
exten => 123,1,Answer()
exten => 123,n,Playback(tt-weasels)
```

### Смотрите также

Background(), ControlPlayback()

## Playtones()

***Воспроизводит набор тонов***

Playtones(*набортонов*)

Воспроизводит набор тонов. После начала воспроизведения тонов выполнение немедленно продолжается со следующего шага. Аргумент *набортонов* – это или имя тона, определенное в конфигурационном файле indications.conf, или заданный список частот и длительностей. Описание спецификации набора тонов приведено в indications.conf.

Для прекращения воспроизведения тонов используется приложение StopPlaytones().

```
; воспроизводить сигнал "занято" в течение двух секунд,
; а затем еще две секунды - сигнал перегрузки линии
exten => 123,1,Playtones(busy)
exten => 123,2,Wait(2)
exten => 123,3,StopPlaytones()
exten => 123,4,Playtones(congestion)
exten => 123,5,Wait(2)
exten => 123,6,StopPlaytones()
exten => 123,7,Goto(1)
```

### Смотрите также

StopPlaytones(), indications.conf, Busy(), Congestion(), Progress(), Ringing()

## PrivacyManager()

***Требует от вызывающего абонента ввода номера телефона, если не получена информация Caller ID***

PrivacyManager([*maxretries*[,*minlength*[,*опции*]]])

Если Caller ID (ID звонящего) не получен, это приложение отвечает каналу и просит вызывающего абонента ввести его номер телефона. По умолчанию абоненту дается три попытки.

PrivacyManager() задает для переменной канала PRIVACYMGRSTATUS значение SUCCESS или FAILURE. Если Caller ID поступает по каналу, PrivacyManager() не выполняет никаких действий.

Если для параметра *опции* задано значение `j` и вызывающий абонент не смог ввести свой номер Caller ID, обработка вызова продолжится в приоритете `n` + 101 (где `n` – текущий приоритет).

Конфигурационный файл privacy.conf меняет функциональность приложения `PrivacyManger()`. Он содержит следующие две строки:

*`maxretries`* (максимум попыток)

Задает максимальное число попыток, которое может сделать вызывающий абонент для ввода номера Caller ID (по умолчанию 3).

*`minlength`* (минимальная длина)

Задает минимально допустимое количество символов во вводимом номере Caller ID (по умолчанию 10).

Настройки *`maxretries`* и *`minlength`* также могут быть переданы в приложение в качестве параметров. Параметры, передаваемые в приложение, переопределяют любые настройки в файле privacy.conf.

```
exten => 123,1,Answer()
exten => 123,n,PrivacyManager()
exten => 123,n,GotoIf($["${PRIVACYMGRSTATUS}" = "FAILURE"]?bad)
exten => 123,n,Dial(Zap/1)
exten => 123,n,Hangup()
exten => 123,n(bad),Playback(im-sorry)
exten => 123,n,Playback(vm-goodbye)
exten => 123,n,Hangup()
```

### Смотрите также

`Zapateller()`

## Progress()

***Служит индикатором хода выполнения вызова***

`Progress()`

Указывает каналу на необходимость предоставить абоненту информацию о ходе выполнения. Каждый тип каналов в Asterisk имеет собственный способ обмена информацией о ходе обработки вызова.

```
; предоставляем информацию о ходе выполнения вызывающему
; каналу, ожидаем 5 с, а затем отвечаем на звонок
exten => 123,1,Progress()
exten => 123,n,Wait(5)
exten => 123,n,Answer()
```

### Смотрите также

`Busy()`, `Congestion()`, `Ringing()`, `Playtones()`

## Queue()

***Помещает текущий вызов в заданную очередь вызовов***

Queue(*имяочереди*[,*опции*[,*URL*[,*переопределитьприветствие*[,*времяожидания*[,*AGI*]]]]])

Помещает входящий вызов в очередь вызовов, заданную аргументом *имяочереди*, соответственно описанию в файле queues.conf.

Аргумент *опции* может содержать нуль или более следующих символов:

d

Устанавливается вызов качества данных (модемный), то есть с минимальной задержкой.

h

Вызываемый абонент получает возможность разорвать соединение нажатием кнопки *.

H

Вызывающий абонент получает возможность разорвать соединение нажатием кнопки *.

i

Запросы на пересылку вызова, поступающие от участников очереди, игнорируются, при поступлении таких запросов никакие действия не выполняются.

n

Повторные попытки по истечении времени ожидания запрещены; происходит выход из этого приложения, и выполнение переходит к следующему шагу.

r

Вместо музыки во время ожидания вызывающий абонент будет слышать гудки.

t

Вызванный абонент получает возможность переадресовывать вызов.

T

Вызывающий абонент получает возможность переадресовывать вызов.

w

Вызванный абонент получает возможность записывать разговор на диск.

W

Вызывающий абонент получает возможность записывать разговор на диск.

Кроме переадресации, вызов может быть припаркован, а затем на него может ответить другой абонент.

Аргумент *переопределитьприветствие* переопределяет стандартное приветствие, воспроизводимое агентам обработки очереди перед тем, как они ответят на указанный вызов.

Необязательный URL будет отправлен вызываемой стороне, если канал поддерживает это.

Соответственно аргументу *времяожидания* вызов будет ожидать ответа в течение заданного промежутка времени, который проверяется между каждым циклом, заданным параметрами *timeout* и *retry* в файле queues.conf. Обработка вызова продолжится со следующего приоритета диалплана.

По завершении это приложение задает переменную канала QUEUESTATUS. Ей может быть задано одно из следующих значений:

TIMEOUT (время ожидания)

Вызов находился в очереди слишком долго, и время ожидания истекло. См. параметр *времяожидания*.

FULL (полная)

Очередь была уже заполнена. См. настройку очереди maxlen в файле queues.conf.

JOINEMPTY (присоединился к пустой)

Вызывающий абонент не мог быть поставлен в очередь, поскольку не было участников, которые могли бы ответить на звонок. См. настройку очереди joinempty в файле queues.conf.

LEAVEEMPTY (покинул пустую)

Вызывающий абонент был поставлен в очередь, но затем все участники обработки очереди покинули ее. См. настройку очереди leavewhenempty в файле queues.conf.

JOINUNAVAIL (присоединился к недоступной)

Вызывающий абонент был поставлен в очередь, но затем все участники обработки очереди стали недоступными. См. настройку очереди leavewhenempty в файле queues.conf.

```
; помещаем вызывающего абонента в очередь techsupport
exten => 123,1,Answer()
exten => 123,2,Queue(techsupport,t)
```

## Смотрите также

AddQueueMember(), RemoveQueueMember(), PauseQueueMember(), UnpauseQueueMember(), AgentLogin(), queues.conf, QUEUE_MEMBER_COUNT, QUEUE_MEMBER_LIST, QUEUE_WAITING_COUNT

# QueueLog()

***Записывает произвольные события очереди в журнал регистрации очереди***

QueueLog(*имяочереди*, *уникальныйid*, *участник*, *событие*[, *дополнительнаяинформация*])

Записывает произвольное событие очереди в журнал регистрации очереди. Параметр *имяочереди* определяет имя очереди вызовов. Параметр *уникальныйid* определяет уникальный идентификатор для канала. Параметр *участник* определяет, к какому участнику обработки очереди относится событие. Для параметров *событие* и *дополнительнаяинформация* произвольно могут быть заданы необходимые данные.

```
; Записываем произвольное событие
; в журнал регистрации очереди
exten => 123,1,QueueLog(myqueue,${UNIQUEID},Agent/123,MyTestEvent)
```

## Смотрите также

Queue()

# Random()

***Выполняет условный переход на основании вероятности***

Random([*вероятность*]:[[*контекст*,]*добавочныйномер*,]*приоритет*)

Это приложение признано устаревшим и заменено приложением GotoIf($[${RAND(1,100)} > *номер*]?*метка*).

Выполняет условный переход в заданный приоритет (и необязательные *добавочныйномер* и *контекст*) на основании вероятности. В качестве вероятности должно использоваться целое число в диапазоне от 1 до 100. Приложение перейдет в заданный приоритет с заданной вероятностью.

```
; снова и снова выбираем случайный номер
exten => 123,1,SayNumber(${RAND(1|10)})
exten => 123,n,Goto(1)
```

## Смотрите также

RAND

# Read()

***Читает DTMF-коды, набираемые вызывающим абонентом, и сохраняет результат в переменной***

Read(*переменная*[,*имяфайла*[,*максимумцифр*[,*опция*[,*попытки*[,*времяожидания*]]]]])

Читает вводимую пользователем строку цифр, оканчивающуюся символом #, в заданную переменную.

Другие аргументы:

*имяфайла*

Определяет файл, воспроизводимый перед чтением цифр.

*максимумцифр*

Задает максимально допустимое количество цифр. Если этот аргумент задан, приложение прекращает чтение после того, как было введено максимально допустимое количество цифр (не требуя от пользователя нажатия кнопки #). Значение по умолчанию – 0 (ограничений нет, ожидает нажатия кнопки #). Любое значение меньше 0 означает то же самое. Максимально допустимое значение – 255.

*опция*

Нуль или более следующих опций:

`s`

Немедленно возвращается, если линия не отвечает.

`i`

Трактует имя файла как настройку тона из файла indications.conf.

`n`

Читает последовательность цифр, даже если нет ответа на вызов.

*попытки*

Если больше 1, то означает количество попыток чтения при условии, что данные не были введены.

*времяожидания*

Если больше 0, это значение переопределит время ожидания по умолчанию.

```
; читаем двузначный номер и воспроизводим
; его вызывающему абоненту
exten => 123,1,Read(NUMBER,,2)
exten => 123,2,SayNumber(${NUMBER})
exten => 123,3,Goto(1)
```

## Смотрите также

`SendDTMF()`

# ReadFile()

***Читает содержимое файла в переменную***

`ReadFile(`*переменная=имяфайла*`,`*длина*`)`

Приложение `ReadFile` фиксирует содержимое файла, определенного аргументом *имяфайла*. Максимальный размер считываемого содержимого определяется аргументом *длина*.

```
; читаем в переменную первые 80 символов файла
exten => 123,1,Answer()
exten => 123,n,ReadFile(TEST=/tmp/test.txt,80)
exten => 123,n,SayAlpha(${TEST})
```

### Смотрите также

System(), Read()

## RealTime()

***Выполняет поиск информации в обработчике конфигурации RealTime***

RealTime(*семейство*, *соответствующийстолбец*, *значение*[, *префикс*])

**Использует систему обработки конфигурации RealTime для чтения данных в переменные канала. Все уникальные имена столбцов (из указанного семейства) будут заданы как переменные канала с добавлением необязательного префикса к имени (например, префикс var_ превратит имя столбца name в переменную ${var_имя}).**

```
; извлекаем все столбцы из таблицы sipfriends, где столбец
; name содержит значение John, и используем для всех
; переменных префикс John_
exten => 123,1,RealTime(sipfriends,name,John,John_)
; теперь прочитаем значение столбца port
exten => 123,n,SayNumber(${John_port})
```

### Смотрите также

RealTimeUpdate()

## RealTimeUpdate()

***Обновляет значение посредством обработчика конфигурации RealTime***

RealTimeUpdate(*семейство*, *соответствующийстолбец*, *значение*, *новыйстолбец*, *новоезначение*)

**Использует систему обработки конфигурации RealTime для обновления значения. В столбце *новыйстолбец* в семействе, соответствующем столбцу *соответствующийстолбец* = *значение*, будет обновлено значение *новоезначение*.**

**В переменной канала REALTIMECOUNT будет сохранено количество обновленных строк или −1 в случае возникновения ошибки.**

```
; зададим для столбца port таблицы sipfriends
; со значением John новое значение, 5061
exten => 123,1,RealTimeUpdate(sipfriends,name,John,port,5061)
```

### Смотрите также

RealTime

# Record()

***Записывает в файл аудиосигнал, передаваемый по каналу***

Record(*имяфайла*.*формат*[,*пауза*[,*максимальнаяпродолжительность*[,*опции*]]])

Записывает аудиосигнал из канала в файл с заданным параметром *имяфайла*. Если файл уже существует, он будет перезаписан.

К необязательным аргументам относятся:

*формат*

Определяет формат, в котором будет записываться файл.

*пауза*

Определяет допустимую паузу (в секундах), после которой запись будет закончена и выполнение перейдет к следующему приоритету диалплана.

*максимальнаяпродолжительность*

Устанавливает максимальную продолжительность записи, в секундах. Если не задана или равна 0, ограничений по продолжительности нет.

*опции*

Может содержать любой из следующих символов:

a

Дописываем в конец файла, а не перезаписываем его.

n

Не отвечаем на звонок, но все равно выполняем запись.

q

Скрытый режим; не воспроизводим звуковой сигнал в начале записи.

s

Не выполняем запись, пока не установлено соединение.

t

Используем альтернативную кнопку завершения * вместо применяемой по умолчанию #.

x

Игнорируем все кнопки завершения и продолжаем запись до разрыва соединения.

Если *имяфайла* содержит %d, эти символы будут замещены номером, увеличивающимся на 1 при каждой записи файла.

Пользователь может нажать кнопку #, чтобы завершить запись и продолжить выполнение со следующего приоритета диалплана.

```
; записываем имя вызывающего абонента
exten => 123,1,Playback(pls-rcrd-name-at-tone)
exten => 123,n,Record(/tmp/name.gsm,3,30)
exten => 123,n,Playback(/tmp/name)
```

# RemoveQueueMember()

***Динамически удаляет участников очереди***

RemoveQueueMember(*имяочереди*[,*интерфейс*[,*опции*]])

Динамически удаляет заданный интерфейс из очереди обработки вызовов, заданной аргументом *имяочереди*. Если интерфейс не задан, это приложение удаляет текущий канал из очереди.

Если параметр *опции* имеет значение j, интерфейса нет в очереди и существует приоритет n + 101 (где n – текущий приоритет), приложение перейдет в этот приоритет.

```
; удалим SIP/3000 из очереди techsupport
exten => 123,1,RemoveQueueMember(techsupport,SIP/3000)
```

## Смотрите также

Queue(), AddQueueMember(), PauseQueueMember(), UnpauseQueueMember()

# ResetCDR()

***Сбрасывает запись параметров вызова***

ResetCDR([*опции*])

Сбрасывает все поля в записи параметров вызовов для текущего канала. Параметр *опции* может содержать нуль или более следующих опций:

a

Сохраняем все записи, помещенные в стек.

w

Сохраняем текущую запись CDR, прежде чем выполнить ее сброс.

v

Сохраняем переменные CDR.

```
; делаем копию текущей записи CDR и сбрасываем ее
exten => 123,1,Answer()
exten => 123,2,Playback(tt-monkeys)
exten => 123,3,ResetCDR(wv)
exten => 123,4,Playback(tt-monkeys)
```

## Смотрите также

ForkCDR(), NoCDR()

# RetryDial()

***Делает попытку позвонить и повторяет ее в случае неудачи***

RetryDial(*объявление*, *ожидание*, *циклы*, *технология/ресурс*[&*технология2/ресурс2*...] [,*времяожидания*][,*опции*][,*URL*])

Делает попытку совершить звонок. Если нет доступного канала, воспроизводится файл, определенный аргументом *объявление*, а затем приложение ожидает заданное аргументом *ожидание* количество секунд и повторяет вызов. Если количество выполненных попыток равно значению, заданному аргументом *циклы*, обработка вызова продолжится в следующем приоритете диалплана. Если аргументу *циклы* задано значение 0, попытки дозвониться будут выполняться бесконечно.

В ходе ожидания может быть введен одноразрядный добавочный номер. Если этот номер существует в любом из контекстов, определенных в переменной ${EXITCONTEXT} (если определена), или в текущем, вызов немедленно будет переадресован на этот добавочный номер.

Все аргументы после аргумента *циклы* передаются непосредственно в приложение Dial().

```
; трижды пытаемся дозвониться на номер через IAX,
; повторяя попытку каждые пять секунд
exten => 123,1,RetryDial(priv-trying,5,3,IAX2/VOIP/8885551212,30)
; если вызывающий абонент в процессе ожидания нажимает 9,
; звоним по этому номеру по каналу Zap/4
exten => 9,1,RetryDial(priv-trying,5,3,Zap/4/8885551212,30)
```

## Смотрите также

Dial()

# Return()

***Возвращает выполнение из Gosub() или GosubIf()***

Return()

Возвращает выполнение из ранее вызванных приложений Gosub() или GosubIf(). Если Gosub() или GosubIf() не вызывались до этого, Return() завершается аварийно.

## Смотрите также

Gosub(), StackPop()

# Ringing()

***Сигнализирует о наличии тонального сигнала вызова***

Ringing()

Указывает каналу передать тональный сигнал вызова абоненту. То, как именно обозначается сигнал вызов, определяет драйвер канала.

Заметьте, что данное приложение не предоставляет фактический звук звонка абоненту. Для этого используется приложение Playtones().

```
; показываем, что телефон звонит,
; даже несмотря на то, что это не так
exten => 123,1,Ringing()
exten => 123,2,Wait(5)
exten => 123,3,Playback(tt-somethingwrong)
```

### Смотрите также

Busy(), Congestion(), Progress(), Playtones()

## SayAlpha()

***Проговаривает строку***

SayAlpha(*строка*)

Проговаривает заданную строку, используя текущую настройку языка для канала. Подробнее о том, как изменить язык текущего канала, рассказывается в описании функции CHANNEL.

```
exten => 123,1,SayAlpha(ABC123XYZ)
```

### Смотрите также

SayDigits(), SayNumber(), SayPhonetic(), CHANNEL

## SayDigits()

***Проговаривает заданные цифры***

SayDigits(*цифры*)

Проговаривает заданные цифры, используя текущую настройку языка для канала. Подробнее о том, как изменить язык текущего канала, рассказывается в описании функции CHANNEL.

```
exten => 123,1,SayDigits(1234)
```

### Смотрите также

SayAlpha(), SayNumber(), SayPhonetic(), CHANNEL

## SayNumber()

***Проговаривает заданный номер***

SayNumber(*цифры*[,*род*])

Проговаривает заданный номер, используя текущую настройку языка для канала. Подробнее о том, как изменить язык текущего канала, рассказывается в описании функции CHANNEL.

Если в текущем языке есть категория рода, для изменения рода проговариваемого номера можно передать аргумент *род*. Могут использоваться следующие значения аргумента *род*:

- `f` для женского рода, `m` для мужского рода и `n` для среднего рода в таких европейских языках, как португальский, французский, испанский и немецкий.
- `c` для общего и `n` для среднего родов в скандинавских языках, таких как датский, шведский и норвежский.
- `p` для многозначных числительных в немецком языке.

```
; проговорим номер по-английски
exten => 123,1,Set(CHANNEL(language)=en)
exten => 123,2,SayNumber(1234)
```

Чтобы это приложение работало не только с английским языком, необходимы соответствующие звуковые файлы для языков, которые вы желаете использовать.

### Смотрите также

`SayAlpha()`, `SayDigits()`, `SayPhonetic()`, `CHANNEL`

## SayPhonetic()

***Проговаривает заданную строку , используя фонетический алфавит***

`SayPhonetic(`*строка*`)`

Проговаривает заданную строку, используя фонетический алфавит, принятый в НАТО.

```
exten => 123,1,SayPhonetic(asterisk)
```

### Смотрите также

`SayAlpha()`, `SayDigits()`, `SayNumber()`

## SayUnixTime()

***Проговаривает указанное время в определенном формате***

`SayUnixTime([`*unixtime*`][,[`*часовойпояс*`][,`*формат*`]])`

Проговаривает указанное время соответственно часовому поясу и формату. Используются аргументы:

*unixtime*

Время, в секундах, прошедшее с 1 января 1970 года. Может быть отрицательным. По умолчанию равно текущему времени.

*часовойпояс*

Часовой пояс. Перечень часовых поясов можно найти по адресу /usr/share/zoneinfo/. По умолчанию используется часовой пояс компьютера.

*формат*

Формат, в котором проговаривается время. Список форматов представлен в файле voicemail.conf. Значение по умолчанию – ABdY 'digits/at' IMp.

```
exten => 123,1,SayUnixTime(,,IMp)
```

### Смотрите также

STRFTIME, STRPTIME, IFTIME

## SendDTMF()

***Посылает произвольную DTMF-последовательность в канал***

SendDTMF(*цифры*[,*времяожидания_мс*])

Передает заданные DTMF-коды в канал. В DTMF-коде допустимы символы 0–9, *, # и A–D. Также может использоваться буква w, которая означает ожидание 500 мс. Аргумент *времяожидания_мс* – это пауза между кодами в миллисекундах. Если не задан, по умолчанию принимается равным 250 мс.

```
exten => 123,1,SendDTMF(3212333w222w366w3212333322321,250)
```

### Смотрите также

Read()

## SendImage()

***Отправляет файл с изображением***

SendImage(*имяфайла*,*опции*)

Отправляет изображение в канал, если поддерживается передача изображений. Это приложение по завершении выполнения задает для переменной канала SENDIMAGESTATUS значение OK или NOSUPPORT (не поддерживается).

Если для параметра *опции* задано значение j, канал не поддерживает передачу изображений и существует приоритет n + 101 (где n – текущий приоритет), выполнение продолжится в нем.

```
exten => 123,1,SendImage(logo.jpg)
```

### Смотрите также

SendText(), SendURL()

## SendText()

***Отправляет текст в канал***

SendText(*текст*,*опции*)

Передает текст в канал, если поддерживается передача текста. По завершении выполнения для переменной канала SENDTEXTSTATUS будет задано одно из следующих значений:

SUCCESS (успех)

Передача текста была успешной.

FAILURE (сбой)

Передать текст не удалось.

NOSUPPORT (не поддерживается)

Базовый канал не поддерживает передачу текста.

Если для параметра *опции* задано значение j, канал не поддерживает передачу текста и существует приоритет n + 101 (где n – текущий приоритет), выполнение продолжится в этом приоритете.

```
exten => 123,1,SendText(Welcome to Asterisk)
```

### Смотрите также

SendImage(), SendURL()

## SendURL()

***Передает заданный URL в канал (если поддерживается)***

SendURL(*URL*[,*опции*])

Предлагает клиенту перейти по заданному URL. Приложение также по завершении задает для переменной SENDURLSTATUS одно из следующих значений:

*SUCCESS*

Передача URL была успешной.

*FAILURE*

Передать URL не удалось.

*NOLOAD*

Базовый канал поддерживает Сеть, но не смог загрузить URL.

*NOSUPPORT*

Базовый канал не поддерживает передачу URL.

Если параметр *опции* содержит значение wait (ожидать), выполнение будет приостановлено до получения подтверждения того, что URL был загружен.

Если для параметра *опции* задано значение j, клиент не поддерживает транспортный протокол HTML и существует приоритет n + 101 (где n – номер текущего приоритета), выполнение продолжится в этом приоритете.

```
exten => 123,1,SendURL(www.asterisk.org,wait)
```

### Смотрите также

SendImage(), SendText()

## Set()

***Присваивает переменной заданное значение***

Set(*n=значение*,[*n2=значение2*...[,*опции*]])

Присваивает переменной *n* заданное значение. Также задает для переменной *n2* значение *значение2*. Если имя переменной начинается с символа _, предполагается одиночное наследование. Если имя переменной начинается с __, предполагается множественное наследование. Наследование используется, когда требуется, чтобы каналы, производные от текущего канала, наследовали переменную текущего канала.

Если для параметра *опции* задано значение g, переменные будут задаваться как глобальные, а не как переменные канала.

```
; задаем значение для переменной DIALTIME, а затем используем ее
exten => 123,1,Set(DIALTIME=20)
exten => 123,1,Dial(Zap/4/5551212,,${DIALTIME})
```

Задание множества переменных и использование опции g признано устаревшим. Пожалуйста, используйте вместо этого несколько вызовов Set() и функцию диалплана GLOBAL().

### Смотрите также

GLOBAL, SET, ENV, channelvariables.txt

## SetAMAFlags()

***Задает AMA-флаги в записи параметров вызовов***

SetAMAFlags(*флаг*)

Задает AMA-флаги в записи параметров вызовов в целях учета вызовов и времени разговора абонента, переопределяя любые настройки AMA конфигурационных файлов канала. Действительными значениями являются default (по умолчанию), omit (не включать), billing (оплачивать) и documentation (документация).

```
exten => 123,1,SetAMAFlags(billing)
```

### Смотрите также

SetCDRUserField(), AppendCDRUserField()

## SetCallerID()

***Задает идентификатор вызывающего абонента для канала***

SetCallerID(*clid*[,*a*])

Это приложение признано устаревшим и заменено приложением Set(CALLERID(all)=*Некоторое Имя <1234>*).

Задает идентификатор вызывающего абонента для канала. Если передается аргумент a, параметру ANI также присваивается заданное значение.

```
; переопределяем Caller ID для этого вызова
exten => 123,1,Set(CALLERID(all)="John Q. Public <8885551212>")
```

### Смотрите также

CALLERID

## SetCallerPres()

***Задает флаги публикации Caller ID***

SetCallerPres(*публикация*)

Задает флаги представления Caller ID для соединения Q.931 PRI.

Действительными представлениями являются:

allowed_not_screened

Публикация разрешена, не экранируется.

allowed_passed_screen

Публикация разрешена, экранирование разрешено.

allowed_failed_screen

Публикация разрешена, экранирование запрещено.

allowed

Публикация разрешена, сетевой номер.

prohib_not_screened

Публикация запрещена, не экранируется.

prohib_passed_screen

Публикация запрещена, экранирование разрешено.

prohib_failed_screen

Публикация запрещена, экранирование запрещено.

prohib

Публикация запрещена, сетевой номер.

*unavailable*

Номер недоступен.

```
exten => 123,1,SetCallerPres(allowed_not_screened)
exten => 123,2,Dial(Zap/g1/8885551212)
```

### Смотрите также

CALLERID()

## SetCDRUserField()

***Задает поле пользователя в записи параметров вызова***

SetCDRUserField(*значение*)

Сохраняет в поле пользователя CDR заданное значение. Поле пользователя CDR – это дополнительное поле, используемое для записи данных, которые не могут быть сохранены в любом другом поле. Записи CDR могут использоваться для целей учета вызовов и времени разговора абонента или для хранения других произвольных данных о конкретном вызове.

```
exten => 123,1,SetCDRUserField(testing)
exten => 123,2,Playback(tt-monkeys)
```

Это приложение признано устаревшим и заменено функцией CDR(). `exten => 123,1,Set(CDR(userfield)=54321)`

### Смотрите также

AppendCDRUserField(), SetAMAFlags()

## SetGlobalVar()

***Присваивает глобальной переменной заданное значение***

SetGlobalVar(*n=значение*)

Это приложение признано устаревшим и заменено приложением Set(GLOBAL(*переменная*)=...).

Присваивает глобальной переменной *n* заданное значение. Глобальные переменные доступны всем каналам.

```
; задаем для глобальной переменной NUMRINGS значение 3
exten => 123,1,SetGlobalVar(NUMRINGS=3)
```

### Смотрите также

Set()

## SetMusicOnHold()

***Устанавливает класс музыки во время ожидания по умолчанию для текущего канала***

SetMusicOnHold(*класс*)

Это приложение признано устаревшим и заменено приложением Set(CHANNEL(классмузыки)=...).

Задает класс по умолчанию музыки во время ожидания для текущего канала. При активации музыки во время ожидания этот класс будет использоваться для выбора воспроизводимой музыки. Классы описаны в конфигурационном файле musiconhold.conf.

```
exten=s,1,Answer()
exten=s,2,SetMusicOnHold(default)
exten=s,3,WaitMusicOnHold()
```

### Смотрите также

`WaitMusicOnHold()`, musiconhold.conf, `MusicOnHold()`

## SetTransferCapability()

***Устанавливает пропускную способность ISDN для канала***

`SetTransferCapability(`*пропускнаяспособность*`)`

Это приложение задает новое значение пропускной способности ISDN для текущего канала. Действительными значениями аргумента *пропускнаяспособность* являются:

`SPEECH`

`0x00`, речь (по умолчанию, голосовые звонки).

`DIGITAL`

`0x08`, неограниченная цифровая информация (вызовы для передачи данных).

`RESTRICTED_DIGITAL`

`0x09`, ограниченная цифровая информация.

`3K1AUDIO`

`0x10`, аудиосигнал частотой 3,1кГц (вызовы для передачи факсов).

`DIGITAL_W_TONES`

`0x11`, неограниченная цифровая информация с тонами/приветствиями.

`VIDEO`

`0x18`, видео.

Это приложение является устаревшим, и его функциональность реализована синтаксисом `Set(CHANNEL(transfercapability)=`*пропускнаяспособность*`)`.

```
exten => 123,1,Set(CHANNEL(transfercapability)=SPEECH)
```

## SIPAddHeader()

***Добавляет SIP-заголовок в исходящий вызов***

`SIPAddHeader(`*Заголовок*`: `*Содержимое*`)`

Добавляет заголовок в SIP-вызов, выполняемый с помощью приложения Dial(). Нестандартный SIP-заголовок должен начинаться с символов X-, например X-Asterisk-Accountcode:. Приложение необходимо использовать с осторожностью, добавление неверных заголовков может стать причиной многих проблем.

Чтобы узнать возможности обеспечения большей гибкости, рассмотрите функцию диалплана SIP_HEADER().

```
exten => 123,1,SIPAddHeader(X-Asterisk-Testing: Just testing!)
exten => 123,2,Dial(SIP/123)
```

### Смотрите также

SIP_HEADER

## SIPDtmfMode()

***Меняет режим передачи DTMF-сигналов для вызова по каналу SIP***

SIPDtmfMode(*режим*)

Меняет режим передачи DTMF-сигналов для вызова по каналу SIP. Допустимые значения аргумента *режим*: inband, info или rfc2833.

```
exten => 123,1,SIPDtmfMode(rfc2833)
exten => 123,2,Dial(SIP/123)
```

### Смотрите также

Приложение A

## SLAStation()

***Станция с поддержкой спаренных линий***

SLAStation(*станция*)

Это приложение должно выполняться SLA-станцией. Формат параметра *станция* зависит от того, как был сформирован вызов. Если просто была снята трубка телефона, параметр *станция* должен содержать только имя станции. Если вызов был инициирован нажатием кнопки линии, имя станции должно сопровождаться символом подчеркивания и именем магистрального канала, связанного с кнопкой этой линии (station1_line2, например).

Более подробную информацию о поддержке спаренных линий можно найти в файле doc/sla.pdf в папке исходного кода Asterisk.

```
exten => 123,1,SLAStation(station1)

        exten => 124,1,SLAStation(station1_line2)
```

### Смотрите также

SLATrunk(), sla.conf

## SLATrunk()

***Магистральный канал с поддержкой спаренных линий***

SLATrunk(*магистральныйканал*)

Это приложение должно выполняться магистральным каналом SLA для входящего вызова. Канал, вызывающий это приложение, должен соответствовать магистральному каналу SLA, заданному параметром *магистральныйканал*.

Более подробную информацию о поддержке спаренных линий можно найти в файле doc/sla.pdf в папке исходного кода Asterisk.

```
exten => 123,1,SLATrunk(line2)
```

### Смотрите также

SLAStation(), sla.conf

## SoftHangup()

***Осуществляет программный разрыв связи по запрашиваемому каналу***

SoftHangup(*технология/ресурс*,*опции*)

Выполняет разъединение заданного канала. Если в качестве аргумента *опции* задана буква a, приложение пытается разъединить все каналы заданного устройства (в настоящее время аргумент *опции* может иметь только это значение).

```
; разъединяем все звонки по Zap/4,
; чтобы мы могли его использовать
exten => 123,1,SoftHangup(Zap/4,a)
exten => 123,2,Wait(2)
exten => 123,3,Dial(Zap/4/5551212)
```

### Смотрите также

Hangup()

## StackPop()

***Удаляет последний адрес из стека Gosub()***

StackPop()

Удаляет последний адрес из стека Gosub(). Часто используется при обработке ошибок в рамках подпрограмм Gosub(), когда больше нет необходимости возвращать управление в точку диалплана, из которой была вызвана подпрограмма Gosub().

```
exten => s,1,Read(input,get-input)
exten => s,n,Gosub(validate,1)
exten => s,n,Dial(SIP/${input})
; Гарантируем ввод номеров в диапазоне от 400 до 499
exten => validate,1,GotoIf($[ ${input} > 499 ]?error,1)
```

```
exten => validate,n,GotoIf($[ ${input} < 400 ]?error,1)
exten => validate,n,Return
exten => error,1,StackPop()
exten => error,2,Goto(s,1)
```

### Смотрите также

`Return()`, `Gosub()`

## StartMusicOnHold()

***Начинает воспроизведение музыки во время ожидания***

`StartMusicOnHold([`*класс*`])`

Воспроизводит музыку во время ожидания, заданную аргументом *класс*, соответственно настройкам в файле musiconhold.conf. Если аргумент опущен, используется класс музыки по умолчанию для канала. Задать класс музыки по умолчанию для канала можно с помощью функции `CHANNEL(`классмузыки`)`.

Возвращается немедленно.

```
; переадресовываем вызовы от систем продаж по телефону
; на этот добавочный номер, чтобы занять их
exten => 123,1,Answer()
exten => 123,2,Playback(tt-allbusy)
exten => 123,3,StartMusicOnHold(default)
exten => 123,4,Wait(600)
exten => 123,5,StopMusicOnHold()
```

### Смотрите также

`WaitMusicOnHold()`, `StopMusicOnHold()`

## StopMixMonitor()

***Прекращает запись разговора по каналу***

`StopMixMonitor()`

Прекращает отслеживание (запись) канала. Это приложение не оказывает никакого воздействия, если запись канала в настоящее время не ведется.

```
exten => 123,1,Answer()
exten => 123,2,MixMonitor(monitor_test.wav)
exten => 123,3,SayDigits(12345678901234567890)
exten => 123,4,StopMixMonitor()
```

### Смотрите также

`MixMonitor()`

## StopMonitor()

***Прекращает запись разговора по каналу***

`StopMonitor()`

Прекращает отслеживание (запись) канала. Это приложение не оказывает никакого воздействия, если запись канала в настоящее время не ведется.

```
exten => 123,1,Answer()
exten => 123,2,Monitor(wav,monitor_test,mb)
exten => 123,3,SayDigits(12345678901234567890)
exten => 123,4,StopMonitor()
```

### Смотрите также

`ChangeMonitor()`

## StopPlaytones()

***Прекращает воспроизведение набора тонов***

`StopPlaytones()`

Прекращает воспроизведение текущего набора тонов.

```
exten => 123,1,Playtones(busy)
exten => 123,2,Wait(2)
exten => 123,3,StopPlaytones()
exten => 123,4,Playtones(congestion)
exten => 123,5,Wait(2)
exten => 123,6,StopPlaytones()
exten => 123,7,Goto(1)
```

### Смотрите также

`Playtones()`, indications.conf

## StopMusicOnHold()

***Прекращает воспроизведение музыки во время ожидания***

`StopMusicOnHold()`

Прекращает воспроизведение музыки во время ожидания по каналу. Если музыка во время ожидания не проигрывалась, не оказывает никакого воздействия.

```
; переадресовываем вызовы от систем продаж по телефону
; на этот добавочный номер, чтобы занять их
exten => 123,1,Answer()
exten => 123,2,Playback(tt-allbusy)
exten => 123,3,StartMusicOnHold(default)
exten => 123,4,Wait(600)
exten => 123,5,StopMusicOnHold()
```

### Смотрите также

WaitMusicOnHold(), StartMusicOnHold()

## System()

***Выполняет команду операционной системы***

System(*команда*)

Выполняет команду в базовой операционной системе. Это приложение задает для переменной канала SYSTEMSTATUS значение FAILURE или SUCCESS в зависимости от того, успешно ли Asterisk выполнила команду.

Это приложение очень похоже на TrySystem(), за исключением того что оно возвратит −1, если не сможет выполнить команду системы, тогда как приложение TrySystem() всегда возвращает 0.

```
exten => 123,1,System(echo hello > /tmp/hello.txt)
```

### Смотрите также

TrySystem()

## Transfer()

***Перенаправляет вызывающего абонента на удаленный добавочный номер***

Transfer([*Технология*/]*вызываемыйномер*[,*опции*])

Указывает, что удаленный вызывающий абонент должен быть переадресован на номер, заданный параметром *вызываемыйномер* (и опционально *Технология*). Если для параметра *Технология* задано значение IAX2, SIP, Zap и т. д., переадресация произойдет, только если входящий вызов поступил по каналу того же типа.

По завершении это приложение задает для переменной канала TRANSFERSTATUS одно из следующих значений:

SUCCESS

Переадресация была успешной.

FAILURE

Переадресация не была успешной.

UNSUPPORTED

Переадресация не поддерживается драйвером базового канала.

Если для параметра *опции* задано значение j, переадресация не поддерживается или не удалась и существует приоритет n + 101 (где n – текущий приоритет), выполнение перейдет в этот приоритет.

```
; переадресовываем звонки с добавочного номера 123
; на добавочный номер SIP/123@otherserver
exten => 123,1,Transfer(SIP/123@otherserver)
```

## TryExec()

***Делает попытку выполнить приложение Asterisk***

`TryExec(`*`приложение`*`(`*`аргументы`*`))`

Делает попытку выполнить заданное приложение Asterisk.

Это приложение очень похоже на приложение `Exec()`, за исключением того что оно всегда возвращается нормально, тогда как приложение `Exec()` будет вести себя так, как если бы базовое приложение было вызвано обычным способом, включая статус выхода. Это приложение может использоваться для перехвата условия, которое при обычном выполнении привело бы к сбою базового приложения.

```
exten => 123,1,TryExec(VMAuthenticate(@default))
```

### Смотрите также

`Exec()`

## TrySystem()

***Делает попытку выполнить команду операционной системы***

`TrySystem(`*`команда`*`)`

Делает попытку выполнить команду в базовой операционной системе. Результат выполнения команды будет помещен в переменную канала `SYSTEMSTATUS`. Это может быть одно из следующих значений:

`FAILURE`

Не удалось выполнить указанную команду.

`SUCCESS`

Указанная команда была выполнена успешно.

`APPERROR`

Указанная команда была выполнена, но возвратила код ошибки.

Это приложение очень похоже на `System()`, за исключением того что всегда завершается нормально, тогда как приложение `System()` будет завершено аварийно, если не сможет выполнить команду системы.

```
exten => 123,1,TrySystem(echo hello > /tmp/hello.txt)
```

### Смотрите также

`System()`

## UnpauseMonitor()

***Возобновляет запись канала***

`UnpauseMonitor()`

Возобновляет запись канала, после того как она была приостановлена с помощью приложения `PauseMonitor()`.

```
exten => 123,1,Answer()
exten => 123,n,Monitor(wav,monitor_test)
exten => 123,n,Playback(demo-congrats)
     ; временно приостанавливаем запись на время сбора секретной информации
exten => 123,n,PauseMonitor()
exten => 123,n,Read(NEWPASS,vm-newpassword)
exten => 123,n,SayDigits(${NEWPASS})
     ; возобновляем и продолжаем запись звонка
exten => 123,n,UnpauseMonitor()
exten => 123,n,Dial(${JOHN})
```

### Смотрите также

Monitor(), StopMonitor(), Page()

## UnpauseQueueMember()

***Возобновление работы участника обработки очереди вызовов***

UnpauseQueueMember([*имяочереди*,]*интерфейс*[,*опции*])

Возобновляет работу участника обработки очереди вызовов (звонки к нему). Является аналогом приложения PauseQueueMember() и действует абсолютно так же, только не приостанавливает, а возобновляет работу данного интерфейса.

По завершении это приложение задает для переменной канала UPQMSTATUS значение UNPAUSED (возобновлен) или NOTFOUND (не найден).

Если для параметра *опции* задано значение j, участник обработки очереди не найден и существует приоритет n + 101 (где n – текущий приоритет), управление вызовов передается в этот приоритет.

```
exten => 123,1,PauseQueueMember(myqueue,SIP/300)
exten => 124,1,UnpauseQueueMember(myqueue,SIP/300)
```

### Смотрите также

PauseQueueMember()

## UserEvent()

***Передает произвольное событие в интерфейс Manager***

UserEvent(*имясобытия*[,*тело*])

Отправляет произвольное событие в интерфейс Manager. В качестве необязательного тела могут быть представлены дополнительные аргументы. Формат события такой:

```
Event: UserEvent
UserEvent: имясобытия
           тело
```

Если параметр *тело* не задан, в событии Manager будут пристутствовать только поля Event и UserEvent.

```
exten => 123,1,UserEvent(BossCalled,${CALLERID(name)} has called the boss!)
exten => 123,2,Dial(${BOSS})
```

### Смотрите также

manager.conf, интерфейс Asterisk Manager

## Verbose()

***Передает произвольный текст в детальный вывод***

Verbose([*уровень*,]*сообщение*)

Посылает заданное сообщение в детальный вывод. В качестве параметра *уровень* должно быть задано целое значение. Если уровень не задан, используется значение по умолчанию, 0.

```
exten => 123,1,Verbose(Somebody called extension 123)
exten => 123,2,Playback(extension)
exten => 123,3,SayDigits(${EXTEN})
```

Необязательный аргумент *уровень* вовсе не является необязательным, если в вызов Verbose() включен разделитель |. Если присутствует разделитель, Verbose() предполагает, что вы собирались задать *уровень* (и отбрасывает все до первого символа |). Поэтому, наверное, лучше выработать привычку всегда задавать параметр *уровень*.

### Смотрите также

NoOp(), Log()

## VMAuthenticate()

***Аутентификация вызывающего абонента по паролям голосовой почты***

VMAuthenticate([*почтовыйящик*][@*контекст*[,*опции*]])

Ведет себя аналогично приложению Authenticate(), за исключением того что используются пароли из файла voicemail.conf.

Если задан параметр *почтовыйящик*, действительными будут считаться только пароли этого почтового ящика. Если параметр *почтовыйящик* не задан, переменной канала AUTH_MAILBOX будет присвоено значение, соответствующее имени почтового ящика, для которого подошел пароль.

Если для параметра *опции* задано значение s, Asterisk пропустит начальные сообщения.

```
; принимает любой пароль почтового ящика из контекста
; голосовой почты default и сообщает нам соответствующий
; номер почтового ящика
exten => 123,1,VMAuthenticate(@default)
exten => 123,2,SayDigits(${AUTH_MAILBOX})
```

### Смотрите также

Authenticate(), voicemail.conf

## VoiceMail()

***Оставляет сообщение голосовой почты в указанном почтовом ящике***

VoiceMail(*почтовыйящик*[@*контекст*][&*почтовыйящик*[@*контекст*]][...]|*опции*)

Оставляет голосовую почту для почтового ящика, заданного параметром *почтовыйящик* (должен быть сконфигурирован в файле voicemail.conf). Если задано более одного почтового ящика, будут использоваться сообщения почтового ящика, заданного первым.

Опции:

*s*

Пропускается воспроизведение инструкций.

*u*

Воспроизводится сообщение о недоступности абонента.

*b*

Воспроизводится сообщение о занятости абонента.

*g(число)*

Увеличивает громкость записи на заданное число децибел (дБ).

*j*

Если запрашиваемый почтовый ящик не существует и имеется приоритет n + 101 (где n – текущий приоритет), этот приоритет будет выполнен следующим.

Если вызывающий абонент во время воспроизведения сообщения нажимает кнопку 0 (нуль), вызов переходит в добавочный номер o (строчная буква o) текущего контекста, если в файле voicemail.conf было задано operator=yes.

Если вызывающий абонент во время воспроизведения сообщения нажимает кнопку *, вызов переходит в добавочный номер a текущего контекста. Это часто используется для направления абонента к личному секретарю.

По завершении это приложение задает переменную канала VMSTATUS. Она будет содержать одно из следующих значений:

SUCCESS

Вызов был успешно отправлен в систему голосовой почты.

USEREXIT

Вызывающий абонент вышел из системы голосовой почты.

FAILED

Система не смогла направить вызов в систему голосовой почты.

```
; отправляем вызывающего абонента в папку голосовой почты, используемую
; в случае недоступности почтового ящика 123
exten => 123,1,VoiceMail(123@default,u)
```

### Смотрите также

`VoiceMailMain()`, voicemail.conf

## VoiceMailMain()

***Выполняет вход в систему голосовой почты***

`VoiceMailMain([`*почтовыйящик*`][@`*контекст*`][,`*опции*`])`

Выполняет вход в основную систему голосовой почты для проверки голосовой почты. Передача аргумента *почтовыйящик* избавит пользователя от необходимости вводить номер почтового ящика. Также необходимо задать аргумент *контекст* для голосовой почты.

Строка *опции* может содержать нуль или более следующих опций:

`s`

Не проводится проверка пароля.

`p`

Эта опция указывает Asterisk трактовать значение *почтовыйящик* как номер, который должен быть добавлен в начало имени почтового ящика, введенного абонентом. Чаще всего это используется, когда на одном сервере Asterisk располагается много ящиков голосовой почты для разных компаний.

`g(`*прирост*`)`

При записи сообщений голосовой почты увеличивает их громкость на величину *прирост*. Эта опция должна быть задана как целое число, обозначающее количество децибел.

`a(`*папка*`)`

Сообщение папки пропускается, и выполняется переход непосредственно к папке, заданной параметром *папка*. Значение по умолчанию для этой опции — `INBOX`.

```
; переходим к меню голосовой почты для почтового
; ящика 123 в контексте голосовой почты default
exten => 123,1,VoiceMailMain(123@default)
```

### Смотрите также

`VoiceMail()`, voicemail.conf

## Wait()

***Ожидает заданное количество секунд***

`Wait(`*количествосекунд*`)`

Ожидает в течение времени, заданного параметром *количествосекунд*. Можно передавать доли секунд. Например, значение *количествосекунд*, равное 1,5, заставит диалплан подождать 1,5 с перед переходом к следующему приоритету.

```
; ожидаем 1,5 с перед воспроизведением сообщения
exten => s,1,Answer()
exten => s,2,Wait(1.5)
exten => s,3,Background(enter-ext-of-person)
```

## WaitExten()

***Ожидает ввода добавочного номера***

`WaitExten([`*количествосекунд*`][,`*опции*`])`

Ожидает ввода нового добавочного номера пользователем заданное количество секунд. Можно задавать доли секунд (например, 1,5 = 1,5 с). Если параметр *количествосекунд* не задан, используется время ожидания ввода добавочного номера по умолчанию. Чаще всего это приложение используется без задания опций *количествосекунд*.

Для параметра *опции* может быть задано такое значение:

`m[(`*класс*`)[`

В процессе ожидания ввода добавочного номера воспроизводится музыка во время ожидания. Можно (необязательно) задать класс музыки во время ожидания в круглых скобках.

```
; ожидаем ввода добавочного номера пользователем
; в течение 15 с
exten => s,1,Answer()
exten => s,2,Playback(enter-ext-of-person)
exten => s,3,WaitExten(15)
```

### Смотрите также

`Background()`, `TIMEOUT`

## WaitForRing()

***Ожидает звонка заданное количество секунд***

`WaitForRing(`*времяожидания*`)`

Ожидает по крайней мере заданное параметром *вермяожидания* количество секунд после завершения следующего звонка.

```
; ожидаем звонка пять секунд, а затем передаем
; некоторую DTMF-последовательность
exten => 123,1,Answer()
exten => 123,2,WaitForRing(5)
exten => 123,3,SendDTMF(1234)
```

### Смотрите также

`WaitForSilence()`

## WaitForSilence()

***Ожидает заданное количество пауз***

WaitForSilence(*необходимаяпауза*[,*повтор*[,*времяожидания*]])

Ожидает заданного параметром *повтор* количества пауз длительностью, в миллисекундах, определенной параметром *необходимаяпауза*. Если параметр *повтор* не задан, приложение ожидает заданного параметром *необходимаяпауза* количества миллисекунд тишины.

Если задана опция *времяожидания*, это приложение возвратится к следующему приоритету диалплана по истечении заданного количества секунд, даже если пауза не была выявлена.

Пожалуйста, используйте опцию *времяожидания* с осторожностью, поскольку она может аннулировать цель применения этого приложения, состоящую в том, чтобы неопределенно долго ожидать тишины в линии. Вероятно, желательно задать очень большое время ожидания, только чтобы избежать бесконечного цикла в случаях, когда тишина не устанавливается никогда.

Это приложение задает для переменной канала WAITSTATUS значение SILENCE или TIMEOUT.

```
; ожидаем трех пауз по 300 мс
exten => 123,WaitForSilence(300,3)
```

### Смотрите также

WaitForRing()

## WaitMusicOnHold()

***Ожидает заданное количество секунд, воспроизводя музыку во время ожидания***

WaitMusicOnHold(*задержка*)

Воспроизводит музыку во время ожидания в течение заданного количества секунд. Если музыка во время ожидания недоступна, задержка все равно будет, но без музыки.

Возвращает 0 по завершении выполнения, или –1 при разрыве соединения.

```
; даем вызывающему абоненту пять минут
; послушать музыку во время ожидания
exten => 123,1,Answer()
exten => 123,2,WaitMusicOnHold(300)
exten => 123,3,Hangup()
```

### Смотрите также

SetMusicOnHold(), musiconhold.conf

# While()

***Начинает выполнение цикла `while`***

`While(`*выражение*`)`

Начинает выполнение цикла `while`. Выполнение возвратится в эту точку, когда будет вызвано приложение `EndWhile()`, если раньше не будет выполнено условие выражения. Если условие выполнено, что приводит к завершению цикла, Asterisk продолжает выполнение диалплана со следующего приоритета после соответствующего приложения `EndWhile()`.

```
exten => 123,1,Set(COUNT=1)
exten => 123,2,While($[ ${COUNT} < 5 ])
exten => 123,3,SayNumber(${COUNT})
exten => 123,4,Set(COUNT=$[${COUNT} + 1])
exten => 123,5,EndWhile()
```

## Смотрите также

`EndWhile()`, `ExitWhile()`, `GotoIf()`

# Zapateller()

***Использует специальный информационный тон для блокирования звонков систем продаж по телефону***

`Zapateller(`*опции*`)`

Генерирует специальный информационный тон для блокирования надоедливых звонков систем продаж по телефону и других автоматизированных звонков.

Аргумент *опции* – это разделенный символами вертикальной черты список опций. Доступны следующие опции:

`answer`

Перед воспроизведением тона должно быть установлено соединение.

`nocallerid`

`Zapateller` воспроизводит тон, только если недоступна информация идентификатора вызывающего абонента.

```
; отвечаем на звонок и воспроизводим SIT-тон,
; если не получаем информации CallerID
exten => 123,1,Zapateller(answer|nocallerid)
```

## Смотрите также

`PrivacyManager()`

# ZapBarge()

***Прослушивает (отслеживает) Zap-канал***

`ZapBarge([`*канал*`])`

Прослушивает заданный Zap-канал или предлагает ввести номер канала, если он не задан. Абоненты, говорящие по каналу, не смогут слышать вас и не получат никакого сигнала о том, что их разговор прослушивается.

Если канал не задан, вам будет предложено ввести номер канала. Введите **4#** для Zap/4, например.

```
exten => 123,1,ZapBarge(Zap/2)
exten => 123,2,Hangup()
```

### Смотрите также

`ZapScan()`

## ZapRAS()

***Выполняет ISDN-сервер удаленного доступа Zaptel***

`ZapRAS(`*аргументы*`)`

Выполняет RAS-сервер ISDN, используя *pppd* для текущего канала. Чтобы использовать эту функцию, канал должен быть выделенным (то есть PRI-источником) и Zaptel-каналом.

Чтобы поддерживать Zaptel, необходимо установить все патчи для pppd. *аргументы* – это разделенный символами вертикальной черты список аргументов.

Это приложение может использоваться только для ISDN-линий, и, чтобы ядро поддерживало `ZapRAS()`, необходимо установить все необходимые патчи. Также ядро должно поддерживать *ppp*.

```
exten => 123,1,Answer()
exten => 123,1,ZapRas(debug|64000|noauth|netmask|255.255.255.0|
10.0.0.1:10.0.0.2)
```

## ZapScan()

***Сканирует Zap-каналы для прослушивания звонков***

`ZapScan([`*группа*`])`

Предоставляет управляющему информационно-справочной службы удобный способ прослушивания Zap-каналов, используя кнопку # для выбора следующего канала и * для выхода. Можно ограничить сканирование конкретной группой каналов (определенной функцией `GROUP()`), задавая аргумент *группа*.

```
exten => 123,1,ZapScan()
```

### Смотрите также

`ZapBarge()`

# C

# Справочник по AGI

## ANSWER

Отвечает на вызов, поступающий по каналу (если соединение еще не установлено, то есть трубка не снята).

Возвращаемые значения:

-1
: Сбой.

0
: Успешное выполнение.

## CHANNEL STATUS

CHANNEL STATUS [*имяканала*]

Запрашивает статус канала, определенного параметром *имяканала* или, если канал не задан, текущего.

Возвращаемые значения:

0
: Канал свободен и доступен.

1
: Канал свободен, но зарезервирован.

2
: Канал подключен.

3

Производится набор номера.

4

По линии поступил вызов.

5

Линия подключена.

6

Линия занята.

## DATABASE DEL

DATABASE DEL *семейство ключ*

Удаляет запись из базы данных Asterisk для заданного семейства и ключа.

Возвращаемые значения:

0

Сбой.

1

Успешное выполнение.

## DATABASE DELTREE

DATABASE DELTREE *семейство* [*деревоключей*]

Удаляет семейство и/или дерево ключей из базы данных Asterisk.

Возвращаемые значения:

0

Сбой.

1

Успешное выполнение.

## DATABASE GET

DATABASE GET *семейство ключ*

Извлекает значение из базы данных Asterisk по заданному семейству и ключу.

Возвращаемые значения:

0

Не задано.

1 ( *значение* )

Значение задано (и приведено в круглых скобках).

## DATABASE PUT

DATABASE PUT *семейство ключ значение*

Добавляет заданное значение для указанного семейства и ключа или обновляет запись базы данных Asterisk.

Возвращаемые значения:

0

Сбой.

1

Успешное выполнение.

## EXEC

EXEC *приложение опции*

Выполняет заданное приложение диалплана, включая опции.

Возвращаемые значения:

-2

Не удалось найти приложение.

*значение*

Возвращаемое значение приложения.

## GET DATA

GET DATA *имяфайла* [*времяожидания*] [*максимум_цифр*]

Воспроизводит аудиофайл, заданный параметром *имяфайла*, и принимает DTMF-последовательность максимальной длиной *максимум_цифр*. Подобно приложению диалплана Background().

Возвращаемое значение:

*значение*

Цифры, полученные от вызывающего абонента.

## GET FULL VARIABLE

GET FULL VARIABLE *имяпеременной* [*имяканала*]

Если переменная, указанная в параметре *имяпеременной*, задана, возвращает ее значение в круглых скобках. Эта команда, в отличие от GET VARIABLE, принимает имена комплексных и встроенных переменных.

Возвращаемые значения:

0

Канал или переменная не заданы.

1 ( *значение* )

Значение получено (и приведено в круглых скобках).

## GET OPTION

GET OPTION *имяфайла коды_выхода* [*времяожидания*]

Ведет себя аналогично STREAM FILE, но имеет опцию *времяожидания* (в секундах).

Возвращаемое значение:

*значение*

ASCII-значение полученных цифр, в десятичной форме.

## GET VARIABLE

GET VARIABLE *имяпеременной*

Если переменная задана, возвращает ее значение в круглых скобках. Эта команда не принимает комплексных или встроенных переменных; используйте команду GET FULL VARIABLE, если приложению необходимы такие типы переменных.

Возвращаемые значения:

0

Канал или переменная не заданы.

1 ( *значение* )

Значение получено (и приведено в круглых скобках).

## HANGUP

HANGUP [*имяканала*]

Выполняет разрыв соединения по указанному каналу или, если канал не задан, по текущему каналу.

Возвращаемые значения:

-1

Указанный канал не существует.

1

Разрыв соединения выполнен успешно.

## NoOp

NoOp [*текст*]

Не выполняет никакого действия. В качестве побочного эффекта это команда выводит значение параметра *текст* в консоль Asterisk. Обычно используется в целях отладки.

Возвращаемое значение:

0

Канал или переменная не заданы.

## RECEIVE CHAR

RECEIVE CHAR *времяожидания*

Принимает один символ текста из канала. Параметр *времяожидания*, в миллисекундах, задает максимальную продолжительность ожидания ввода; при значении 0 ожидание будет длиться неопределенно долго. Обратите внимание, что большинство каналов не поддерживают прием текста.

Возвращаемые значения:

-1 (разрыв)

Сбой или разрыв соединения.

*char* (времяожидания)

Время ожидания.

*значение*

ASCII-значение символа, в десятичной форме.

## RECORD FILE

RECORD FILE *имяфайла формат коды_выхода времяожидания* [*фрагменты_смещения*] [BEEP] [*s=пауза*]

Записывает аудиосигнал, передаваемый по каналу, в заданный файл до получения определенного кода выхода (DTMF). Аргумент *формат* определяет тип записываемого файла (wav, gsm и пр.). Аргумент *времяожидания* – это максимальная продолжительность записи в миллисекундах. Может быть задан равным –1, что означает отсутствие времени ожидания. Аргумент *фрагменты_смещения* необязательный; если задан, запись начнется со смещением на заданное количество фрагментов, без выхода за пределы файла. Аргумент BEEP обусловит подачу звукового сигнала абоненту для обозначения начала операции записи. Аргумент *пауза* – это допустимая пауза, в секундах, после которой функция возвращается, даже если не поступили DTMF-коды или не истекло время ожидания. Значению паузы должна предшествовать запись s=. Этот аргумент также необязательный.

Возвращаемые значения:

-1

Сбой.

0

Успешная запись.

## SAY ALPHA

SAY ALPHA *номер коды_выхода*

Проговаривает заданную строку символов, возвращаясь досрочно в случае получения по каналу заданных DTMF-кодов.

Возвращаемые значения:

-1

Ошибка или разрыв соединения.

0

Воспроизведение завершено без прерывания кодом выхода.

*значение*

ASCII-значение цифры (если введена), в десятичной форме.

## SAY DATE

SAY DATE *дата коды_выхода*

Проговаривает дату, заданную аргументом *дата*, возвращаясь досрочно в случае получения по каналу заданных DTMF-кодов. *дата* – это количество секунд, прошедших с 00:00:00 по Гринвичу (Coordinated Universal Time, UTC) 1 января 1970 года.

Возвращаемые значения:

-1

Ошибка или разрыв соединения.

0

Воспроизведение завершено без прерывания кодом выхода.

*значение*

ASCII-значение цифры (если введена), в десятичной форме.

## SAY DATETIME

SAY DATETIME *датавремя коды_выхода* [*формат*] [*часовойпояс*]

Проговаривает дату и время, заданные аргументом *датавремя*, возвращаясь досрочно в случае получения по каналу заданных DTMF-кодов. *датавремя* – это количество секунд, прошедших с 00:00:00 по Гринвичу 1 января 1970 года. Необязательный аргумент *формат* – это формат, в котором должно проговариваться время. (Полное описание опций формата можно найти в файле voicemail.conf.) По умолчанию *формат* принимает значение ABdY 'digits/at' IMp. Допустимые значения аргумента *часовойпояс* можно найти по адресу /usr/share/zoneinfo/. По умолчанию значение агрумента *часовойпояс* принимается равным часовому поясу по умолчанию сервера Asterisk.

Возвращаемые значения:

-1

Ошибка или разрыв соединения.

0

Воспроизведение завершено без прерывания кодом выхода.

*значение*

ASCII-значение цифры (если введена), в десятичной форме.

## SAY DIGITS

SAY DIGITS *номер коды_выхода*

Проговаривает заданную строку цифр, возвращаясь досрочно в случае получения по каналу заданных DTMF-кодов.

Возвращаемые значения:

-1

Ошибка или разрыв соединения.

0

Воспроизведение завершено без прерывания кодом выхода.

*значение*

ASCII-значение цифры (если введена), в десятичной форме.

## SAY NUMBER

SAY NUMBER *номер коды_выхода*

Проговаривает заданный номер, возвращаясь досрочно в случае получения по каналу любого из заданных DTMF-кодов.

Возвращаемые значения:

-1

Ошибка или разрыв соединения.

0

Воспроизведение завершено без прерывания кодом выхода.

*значение*

ASCII-значение цифры (если введена), в десятичной форме.

## SAY PHONETIC

SAY PHONETIC *строка коды_выхода*

Проговаривает заданную строку символов, используя фонетический алфавит. Возвращается досрочно в случае получения по каналу любого из заданных DTMF-кодов.

Возвращаемые значения:

-1

Ошибка или разрыв соединения.

0

Воспроизведение завершено без прерывания кодом выхода.

*значение*

ASCII-значение цифры (если введена), в десятичной форме.

## SAY TIME

SAY TIME *время коды_выхода*

Проговаривает время, указанное параметром *время*, возвращаясь досрочно в случае получения по каналу любого из заданных DTMF-кодов. *время* – это количество секунд, прошедших с 00:00:00 по Гринвичу 1 января 1970 года.

Возвращаемые значения:

-1

Ошибка или разрыв соединения.

0

Воспроизведение завершено без прерывания кодом выхода.

*значение*

ASCII-значение цифры (если введена), в десятичной форме.

## SEND IMAGE

SEND IMAGE *изображение*

Передает заданное изображение по текущему каналу. Большинство каналов не поддерживают передачу изображений. Имена файлов изображений не должны включать расширения.

Возвращаемые значения:

-1

Ошибка или разрыв соединения.

0

Изображение отправлено или канал не поддерживает отправку изображения.

## SEND TEXT

SEND TEXT *"текст_для_отправки"*

Передает указанный текст по текущему каналу. Большинство каналов не поддерживают передачу текста. Текст, содержащий более одного слова, должен быть заключен в кавычки, поскольку команда принимает только один аргумент.

Возвращаемые значения:

-1

Ошибка или разрыв соединения.

0

Текст отправлен или канал не поддерживает отправку текста.

## SET AUTOHANGUP

SET AUTOHANGUP *время*

Обусловливает автоматическое разъединение связи по каналу по истечении количества секунд, заданного параметром *время*. Конечно, разрыв может произойти и раньше. Задавая для параметра *время* значение 0, можно отключить возможность autohangup (автоматическое разъединение) для данного канала.

Возвращаемое значение:

0

Возможность автоматического разъединения установлена.

## SET CALLERID

SET CALLERID *номер*

Меняет идентификатор вызывающего абонента текущего канала.

Возвращаемое значение:

1

Идентификатор вызывающего абонента задан.

## SET CONTEXT

SET CONTEXT *контекст*

Задает *контекст*, в котором будет продолжаться выполнение после выхода из приложения AGI.

Возвращаемое значение:

0

Контекст задан.

## SET EXTENSION

SET EXTENSION *добавочныйномер*

Меняет *добавочныйномер*, в который перейдет выполнение после выхода из приложения AGI.

Возвращаемое значение:

0

Добавочный номер задан.

## SET MUSIC ON

SET MUSIC ON [on|off] [*класс*]

Активирует/деактивирует генератор музыки во время ожидания. Если *класс* не задан, будет использоваться класс музыки во время ожидания по умолчанию.

Возвращаемое значение:

0

Всегда возвращает 0.

## SET PRIORITY

SET PRIORITY *приоритет*

Меняет приоритет для продолжения выполнения после выхода из приложения AGI. Значение аргумента *приоритет* должен быть действительным приоритетом или меткой.

Возвращаемое значение:

0

Приоритет задан.

## SET VARIABLE

SET VARIABLE *имяпеременной значение*

Задает или обновляет *значение* переменной, заданной параметром *имяпеременной*. Если переменной не существует, она создается.

Возвращаемое значение:

1

Переменная задана.

## STREAM FILE

STREAM FILE *имяфайла коды_выхода* [*фрагмент_смещения*]

Воспроизводит аудиофайл, определенный параметром *имяфайла*. При этом допускается прерывание воспроизведения путем введения кодов, определенных параметром *коды_выхода*, если таковые заданы. Если вы желаете запретить любые коды выхода, введите вместо параметра *коды_выхода* двойные кавычки. Если предоставлен параметр *фрагмент_смещения*, воспроизведение начнется с фрагмента, заданного этим параметром. Помните, не надо включать расширение файла в параметр *имяфайла*.

Возвращаемые значения:

0

Воспроизведение завершено, коды введены не были.

-1

Ошибка или разрыв соединения.

*значение*

ASCII-значение цифры (если введена), в десятичной форме.

## TDD MODE

Активирует и отключает возможность взаимодействия со слуховыми аппаратами (Telecommunications Devices for the Deaf, TDD) для этого канала.

Возвращаемые значения:

0

Канал не поддерживает TDD.

1

Успешное выполнение.

## VERBOSE

VERBOSE *сообщение уровень*

Отправляет сообщение, определенное аргументом *сообщение*, в консоль через систему детальных сообщений. Аргумент *уровень* – это минимальный уровень детальности, при котором сообщение появится в интерфейсе командной строки Asterisk.

Возвращаемое значение:

0

Всегда возвращает 0.

## WAIT FOR DIGIT

WAIT FOR DIGIT *времяожидания*

Ожидает введения DTMF-кода в течение количества секунд, заданного для канала параметром *времяожидания*. Используйте –1 в качестве значения параметра *времяожидания*, если вы хотите блокировать вызов неопределенно долго.

Возвращаемые значения:

-1

Ошибка или сбой канала.

0

Время ожидания истекло.

*значение*

ASCII-значение цифры (если введена), в десятичной форме.

# D

# Конфигурационные файлы

Данное приложение является справочником по конфигурационным файлам, которые не были рассмотрены в предыдущих приложениях. Настройки VoIP-канала можно найти в приложении A. Справочные данные по диалплану приведены в приложении B.

Для каждого модуля Asterisk, который предполагается использовать, необходим конфигурационный файл. Эти файлы, .conf, содержат определения каналов, описывают внутренние сервисы, определяют местоположения других модулей или устанавливают связь с диалпланом. Для получения функциональной системы необязательно настраивать их все, а только те, которые необходимы для вашей конфигурации. Asterisk поставляется с образцами всех конфигурационных файлов, но Asterisk можно запустить, не имея ни одного из них. В этом случае система работать не будет, но это наглядно демонстрирует модульный принцип платформы.

Не найдя ни одного файла .conf, Asterisk примет некоторые решения относительно модулей. Например, всегда выполняются следующие шаги:

- Загружается журнал регистрации событий Asterisk, и события протоколируются в файле /var/log/asterisk/event_log.
- Регистрируются команды интерфейса Manager.
- Запускается ядро офисной АТС.
- Выделяется ряд RTP-портов от 5000 до 31000.

- Загружается несколько встроенных приложений, таких как `Answer()`, `Background()`, `GotoIf()`, `NoOp()` и `Set()`.
- Запускается динамический загрузчик, то есть механизм, ответственный за загрузку модулей, описанных в файле modules.conf.

Это приложение начинается с детального обсуждения конфигурационного файла modules.conf. Затем кратко рассматриваются другие файлы, которые, возможно, вам придется сконфигурировать для своей системы Asterisk.

# modules.conf

Файл modules.conf управляет тем, какие модули загружаются или не загружаются при запуске Asterisk. Описывается это с помощью структур `load =>` или `noload =>`.

Этот файл является ключевым компонентом для построения безопасной системы Asterisk: лучшей практикой считается загружать только необходимые модули.

Файл modules.conf всегда начинается с заголовка `[modules]` (модули). Asterisk может автоматически загружать все модули, содержащиеся в папке модулей, используя выражение `autoload` (автозагрузка), или загружать только те модули, которые указаны выражениями `load =>`. Мы рекомендуем вручную загружать только необходимые модули, но многие считают, что проще позволить Asterisk автоматически загрузить все, что она найдет в папке /usr/lib/asterisk/modules. После этого определенные модули можно исключить[1] из загрузки с помощью выражений `noload =>`.

Вот пример файла modules.conf:

```
[modules]
autoload=no ; задайте значение yes - и Asterisk будет
            ; загружать все модули, которые найдет
            ; в папке /usr/lib/asterisk/modules

load => res_adsi.so
load => pbx_config.so  ; Требует: ничего
load => chan_iax2.so   ; Требует: res_crypto.so,
                       ; res_features.so
load => chan_sip.so    ; Требует: res_features.so
load => codec_alaw.so  ; Требует: ничего
load => codec_gsm.so   ; Требует: ничего
load => codec_ulaw.so  ; Требует: ничего
load => format_gsm.so  ; Требует: ничего
```

[1] С появлением новой системы выбора компонентов сборки следовать лучшей практике необязательно, если в первую очередь выполняется сборка только необходимых модулей.

```
load => app_dial.so     ; Требует: res_features.so,
                        ; res_musiconhold.so
```

Поскольку мы предполагаем, что сборка Asterisk выполнена в Linux, имена всех используемых модулей заканчиваются расширением .so. Однако если вы выполнили сборку Asterisk в другой операционной системе, расширения будут другими.

На момент написания данной книги существует восемь типов модулей: *ресурсы*, *приложения*, *коннекторы баз данных для хранения Call Detail Record*, *каналы*, *кодеки*, *форматы*, *модули ATC* и *самостоятельные функции*. Остановимся на каждом из них в отдельности.

## adsi.conf

Интерфейс сервисов для аналогового дисплея (Analog Display Services Interface, ADSI) был разработан для того, чтобы телефонные компании могли поставлять улучшенные сервисы по аналоговым телефонным сетям. В Asterisk этот файл можно использовать для передачи команд ADSI в поддерживающие его телефоны. Пожалуйста, обратите внимание, что телефон должен быть напрямую подключен к каналу Zapata. ADSI-сообщения не могут передаваться на удаленный аналоговый телефон через VoIP-соединение.

Приложение `Voicemail()` требует наличия модуля res_adsi.so; однако необязательно, чтобы использовался файл adsi.conf. Информация об ADSI не является общедоступной. Документацию необходимо покупать у компании Telcordia.

## adtranvofr.conf

До появления технологии Voice over IP некоторое время в качестве средства пакетной передачи голоса широко использовалась технология Voice over Frame Relay (VoFR – передача голоса по сетям с ретрансляцией кадров). Использование оборудования Adtran для поддержки VoFR – часть истории Asterisk.

Однако этот протокол больше не пользуется популярностью в сообществе, поэтому могут возникнуть трудности с его поддержкой.

## agents.conf

Этот файл позволяет создавать агентов вашего центра обработки вызовов и управлять ими. Если используется приложение `Queue()`, можно сконфигурировать настройки агентов для обработки очереди вызовов. Для настройки драйвера канала `AGENT` (агент) используется файл agents.conf.

Раздел [general] файла agents.conf в настоящее время содержит только два параметра. Параметр persistentagents (постоянно хранимые агенты) указывает Asterisk, должен ли сохраняться статус агентов, использующих возможность обратного вызова очередей, в локальной базе данных Asterisk. Если задано значение yes, регистрация удаленного агента будет сохраняться при перезагрузках системы (если только не будет удалена из базы данных какими-либо иными средствами). Параметр multiplelogin (множественная регистрация) говорит Asterisk о возможности регистрации множества агентов с одного добавочного номера.

Следующие параметры, заданные в разделе [agents] (агенты), используются для описания агентов и способа взаимодействия системы с ними. Настройки применяются ко всем агентам, но могут быть переопределены для отдельного агента его индивидуальными настройками:

maxlogintries

Максимальное число попыток, которое дается агенту, чтобы зарегистрироваться. Значение по умолчанию – 3.

autologoff

Принимает аргумент (в секундах), определяющий время ожидания ответа на вызов по каналу агента, по истечении которого агент будет признан недоступным и выгружен из системы.

autologoffunavail

Задается, чтобы выполнялась автоматическая выгрузка агентов из системы в случае возвращения приложением Dial() статуса CHANUNAVAIL в результате попытки дозвониться до этого агента. Значение по умолчанию – no.

ackcall

Принимает аргументы yes и no. Если задано значение yes, агент с возможностью обратного вызова должен после регистрации подтверждать вход в систему нажатием кнопки #. Используется в сочетании с приложением AgentCallbackLogin().

endcall

Если задано значение yes, агент может разрывать соединение нажатием кнопки *. Значение по умолчанию – yes. Задайте no, чтобы Asterisk не выполняла никаких действий при нажатии кнопки * агентом.

wrapuptime

Можно задать этот параметр, чтобы обеспечить агентам некоторую паузу перед передачей им следующего вызова после завершения предыдущего. Эта настройка задается в миллисекундах.

musiconhold => *класс*

Принимает в качестве аргумента класс музыки во время ожидания. Эта настройка применяется ко всем агентам.

`agentgoodbye`

Определяет стандартный звуковой сигнал завершения сеанса для агентов.

`updatecdr`

Принимает аргументы `yes` и `no`. Определяет, должны ли в записях CDR для канала, с которого поступил вызов, быть указаны `agent/agent_id` для обозначения агентов, выполнивших вызовы.

`group`

Определяет группы, в которые входит агент. Группы задаются целыми числами. Если агент входит в несколько групп, их номера (целые числа) перечисляются через запятую.

`recordagentcalls`

Принимает аргументы `yes` и `no`. Определяет, должны ли записываться звонки агента.

`recordformat`

Определяет формат записываемых файлов. Допустимые значения – `wav`, `gsm` или `wav49`. Формат записи по умолчанию – `wav`.

`urlprefix`

В качестве аргумента принимает строку. Строка может быть сформирована как URL и прикрепляться в начало текста, который должен быть добавлен к имени записи.

`savecallsin`

В качестве аргумента принимает путь в файловой системе. Позволяет по своему усмотрению переопределять путь по умолчанию /var/spool/asterisk/monitor/.

Поскольку для хранения вызовов потребуется большое количество дискового пространства, вы захотите определить стратегию хранения этих записей и работы с ними.

Данная папка, вероятно, должна размещаться на отдельном диске, имеющем высокие характеристики производительности.

`custom_beep`

В качестве аргумента принимает имя файла. Может использоваться для определения специального тона уведомления, который будет сообщать всегда подключенному агенту о поступлении входящего вызова.

Последний параметр используется для определения агентов. Как и в файле zapata.conf, параметры конфигурации наследуются от описания `agent =>`, располагающегося выше. Для описания агентов используется следующий формат:

```
agent => id_агента,пароль_агента,имя
```

Например, агента Happy Tempura с идентификатором агента 1000 и паролем 1234 можно определить следующим образом:

```
agent => 1000,1234,Happy Tempura
```

Не забывайте, что файл agents.conf является дополнением к конфигурации очереди вызовов. Самый важный конфигурационный файл очередей вызовов – queues.conf. Без agents.conf можно настроить очередь вызовов только с самой базовой функциональностью.

## alarmreceiver.conf

Приложение `AlarmReceiver()` не утверждено Лабораторией по технике безопасности США (Underwriter's Laboratory, UL) и не должно использоваться как основное или единственное средство получения тревожных сообщений или событий. Надежность этого приложения не гарантируется, поэтому не полагайтесь на него без всестороннего тестирования. Использование этого приложения без тестирования может подвергнуть риску вашу жизнь и/или собственность.

Файл alarmreceiver.conf используется приложением `AlarmReceiver()`, благодаря которому Asterisk может принимать сигналы тревоги по протоколу SIA (Ademco) Contact ID. При поступлении с панели сигнализации вызов должен быть направлен в контекст, вызывающий приложение `AlarmReceiver()`. В свою очередь, `AlarmReceiver()` прочитает конфигурационный файл alarmreceiver.conf и выполнит заданные действия. Все параметры задаются под заголовком `[general]`.

Образец конфигурационного файла будет содержать текущие настройки для данного приложения и очень хорошо задокументирован.

## alsa.conf

Файл alsa.conf используется для конфигурации Asterisk при использовании Advanced Linux Sound Architecture (ALSA – расширенная звуковая архитектура Linux) для обеспечения доступа к звуковой карте, если необходимо. Этот файл можно использовать для конфигурации канала `CONSOLE` (консоль), который чаще всего применяется для создания системы объявлений по всем линиям (хотя, как с любым другим каналом, применив творческий подход, ему можно найти множество разных употреблений). Помните, что полезность ALSA-канала самого по себе ограничена из-за отсутствия пользовательского интерфейса[1].

[1] Да, мы знаем, что пользовательский интерфейс для интерфейса канала – это CLI Asterisk; однако он не может использоваться как телефон и поэтому не является интерфейсом с точки зрения пользователя телефона.

# amd.conf

Это конфигурация для приложения выявления автоответчика в Asterisk, называемого AMD(). Данный файл служит для настройки различных параметров, используемых при выявлении автоответчика на основании таких показателей, как начальная пауза, длительность приветствия, пауза после приветствия и т. д.

# asterisk.conf

Файл asterisk.conf определяет местоположение конфигурационных файлов, папки очереди и модулей, а также папки для записи файлов журнала. Рекомендуется использовать настройки по умолчанию, если вы недостаточно хорошо понимаете последствия их изменения. Файл asterisk.conf формируется автоматически при выполнении команды make samples исходя из собранной ею информации о системе. Он включает раздел [directories] (папки):

```
[directories]
astetcdir => /etc/asterisk
astmoddir => /usr/lib/asterisk/modules
astvarlibdir => /var/lib/asterisk
astdatadir => /var/lib/asterisk
astagidir => /var/lib/asterisk/agi-bin
astspooldir => /var/spool/asterisk
astrundir => /var/run
astlogdir => /var/log/asterisk
```

Кроме того, можно задать раздел [options] (опции), что позволит определить опции запуска (ключи командной строки) в конфигурационном файле. В следующем примере показаны доступные опции и соответствующие им ключи командной строки:

```
[options]
;Под заголовком options можно вводить конфигурационные
;опции, которые также возможно задать с помощью
;командной строки

verbose = 0             ; Уровень детальности для
                        ; протоколирования (-v)
debug = 3               ; Отладка: No или значение (1-4)
nofork=yes | no         ; Выполнение в фоновом режиме
                        ; выключено (-f)
alwaysfork=yes | no     ; Всегда в фоновом режиме, даже
                        ; с -v или -d (-F)
console= yes | no       ; Режим консоли (-c)
highpriority = yes | no ; Выполнение с высоким приоритетом (-p)
initcrypto = yes | no   ; Инициализировать шифрование при
                        ; запуске (-i)
nocolor = yes | no      ; Отключить ANSI-цвета (-n)
dumpcore = yes | no     ; Выполнить дамп ядра при сбое (-g)
```

```
quiet = yes | no        ; Запуск в скрытом режиме (-q)
timestamp = yes | no    ; Активировать временные метки
                        ; в детальном выводе CLI (-T)
runuser = asterisk      ; Пользователь, под учетной
                        ; записью которого выполняется
                        ; asterisk (-U). ПРИМЕЧАНИЕ:
                        ; потребует изменения прав
                        ; доступа к папкам и устройствам
rungroup = asterisk     ; Группа, под учетной записью
                        ; которой выполняется asterisk (-G)
internal_timing = yes | no ; Активировать поддержку
                           ; внутренней синхронизации (-I)
                           ; Эти опции не имеют
                           ; эквивалентных ключей
                           ; командной строки
cache_record_files = yes | no ; Кэшировать файлы,
                              ; создаваемые record(),
                              ; в другой папке
                              ; до завершения записи
record_cache_dir = <dir>
transcode_via_sln = yes | no  ; Определить перекодировку
                              ; через SLINEAR
transmit_silence_during_record = yes | no ; передавать
                                          ; тишину SLINEAR
                                          ; во время
                                          ; записи канала
maxload = 1.0             ; Максимальная средняя нагрузка
                          ; для приема вызовов
maxcalls = 255            ; Максимальное число
                          ; одновременных вызовов, которое
                          ; вы желаете разрешить
execincludes = yes | no   ; Разрешить записи #exec
                          ; в конфигурационных файлах
dontwarn = yes | no       ; Не заваливайте лишней
                          ; информацией системного
                          ; администратора Asterisk,
                          ; он гуру
systemname = <a_string>   ; Имя системы. Используется как
                          ; префикс uniqueid CDR и для
                          ; заполнения ${SYSTEMNAME}
languageprefix = yes | no ; Должен ли код языка быть
                          ; последним или первым
                          ; компонентом имени звукового
                          ; файла? Если выключен, поиск
                          ; звуковых файлов ведется
                          ; в формате <путь>/<язык>/<файл>
                          ; Если включен, поиск ведется
                          ; в формате <язык>/<путь>/<файл>
                          ; (используются только
                          ; относительные пути звуковых
                          ; файлов)
```

# cdr.conf

Файл cdr.conf используется для активации протоколирования записей параметров вызовов в плоский файл или базу данных. Хранение записей вызовов полезно для всевозможных целей, включая учет вызовов и времени разговора абонента, предотвращение мошенничества, оценку QoS и многое другое. Файл cdr.conf содержит некоторые общие параметры, не относящиеся к конкретной базе данных, а, скорее, указывающие, как Asterisk должна обрабатывать передачу информации в базу данных. Вот полный список опций раздела `[general]` файла cdr.conf:

`enable`

Принимает аргументы `yes` и `no`. Определяет, выполняется или нет протоколирование CDR. Если задано значение `no`, это переопределит любой явно загруженный модуль CDR. Значение по умолчанию – `yes`.

`batch`

Принимает аргументы `yes` и `no`. Позволяет Asterisk в конце каждого вызова записывать данные в буфер, а не в базу данных, чтобы сократить нагрузку на систему.

Заметьте, что, если для этой опции задано значение `yes`, в случае неожиданного сбоя системы данные могут быть утрачены.

`size`

Устанавливает максимальное количество записей CDR, накапливаемых в буфере перед передачей на серверную часть систем хранения CDR. Эта настройка имеет смысл, только если для опции `batch` задано значение `yes`. Значение по умолчанию – 100 записей.

`time`

В качестве аргумента принимает целое число (количество секунд). Определяет, через какое время (в секундах) Asterisk очищает буфер и записывает CDR в базу данных, независимо от количества записей в буфере (что определено параметром `size`). Значение по умолчанию – 300 с (5 мин).

`scheduleronly`

Принимает аргументы `yes` и `no`. Если в системе создается и передается в удаленную базу данных большой объем CDR, может быть полезно задать для `scheduleronly` значение `yes`. Поскольку планировщик задач не может начать новую задачу, не завершив текущую, медленная запись CDR может неблагоприятно повлиять на остальные процессы, использующие планировщик. Эта настройка будет указывать Asterisk обрабатывать запись CDR в новом потоке, по сути, назначая специальный планировщик для этой функции. При нормальной работе это обеспечит очень незначительное преимущество.

safeshutdown

Принимает аргументы yes и no. Задание для safeshutdown значения yes не даст Asterisk полностью выключиться, пока буфер не будет полностью очищен и вся информация не будет записана в базу данных. Если для этого параметра задано значение no и происходит выключение Asterisk при наличии информации в буферах, эта информация, скорее всего, будет утрачена.

endbeforehexten

Обычно записи CDR не закрываются до тех пор, пока не будет завершено выполнение всех добавочных номеров. Если эта опция активирована, CDR будет завершаться перед выполнением добавочного номера h, так что такие значения CDR, как end и billsec, могут быть извлечены в этом добавочном номере. Значение по умолчанию – no.

Оставшаяся часть cdr.conf посвящена настройкам нескольких серверных механизмов хранения CDR. Для получения более подробной информации обратитесь к шаблону файла cdr.conf.

# cdr_manager.conf

Файл cdr_manager.conf содержит только заголовок [general] и единственную опцию, enabled, с помощью которой можно определить, должен ли Asterisk Manager API формировать события CDR. Если требуется формировать события CDR, в файле cdr_manager.conf должны присутствовать следующие строки:

```
[general]
enabled=yes
```

Тогда Manager API будет создавать CDR-события, содержащие следующие поля:

```
Event: Cdr
AccountCode:
Source:
Destination:
DestinationContext:
CallerID:
Channel:
DestinationChannel:
LastApplication:
LastData:
StartTime:
AnswerTime:
EndTime:
Duration:
BillableSeconds:
Disposition:
AMAFlags:
UniqueID:
UserField:
```

## cdr_odbc.conf

Asterisk может хранить данные CDR в локальной или удаленной базе данных посредством интерфейса ODBC. Файл cdr_odbc.conf содержит информацию, необходимую Asterisk для соединения с базой данных. Модуль cdr_odbc.so будет пытаться загрузить файл cdr_odbc.conf, и в случае обнаружения информации для соединения с базой данных данные CDR будут записываться туда.

Если вы собираетесь использовать базу данных для хранения данных CDR, придется выбрать одну базу данных из нескольких доступных. Asterisk не любит, когда приходится соединяться с несколькими базами данных CDR, поэтому в папке конфигурации Asterisk не должно быть лишних файлов cdr_<*механизм БД*>.conf.

## cdr_pgsql.conf

Asterisk может хранить данные CDR в базе данных PostgreSQL с помощью модуля cdr_pgsql.so. При загрузке этого модуля необходимая информация считывается из файла cdr_pgsql.conf, и Asterisk устанавливает соединение с базой данных для записи и хранения данных CDR.

## cdr_tds.conf

Asterisk также может хранить данные CDR в базе данных FreeTDS (включая СУБД MS SQL), используя модуль cdr_tds.so. При загрузке этого модуля используется конфигурационный файл cdr_tds.conf. После успешного установления соединения данные CDR записываются в эту базу данных.

## codecs.conf

У большинства кодеков нет настраиваемых параметров; они просто такие, какие они есть.

Однако некоторые кодеки могут вести себя по-разному. Это, главным образом, означает, что они могут быть оптимизированы для конкретной цели, такой как сокращение задержки, оптимизация использования сети или, возможно, обеспечение высококачественного звука.

Файл codecs.conf появился в Asterisk достаточно недавно и на момент написания данной книги обеспечивал возможность конфигурации только параметров Speex. Эти настройки не требуют никаких объяснений, если вы знакомы с протоколом Speex (*http://www.speex.org*).

codecs.conf также позволяет конфигурировать сокрытие потерянных пакетов (Packet Loss Concealment, PLC). Вам необходимо определить раздел `[plc]` и указать `genericplc => true`. Это заставит Asterisk пытать-

ся интерполировать потерянные пакеты. (Активация этой функциональности приведет к небольшому снижению производительности.)

## dnsmgr.conf

Этот файл используется для настройки DNS-поиска: должна ли Asterisk выполнять его регулярно и как часто этот поиск должен осуществляться.

## dundi.conf

Протокол DUNDi используется для динамического поиска VoIP-адреса телефонного номера в сети и для соединения с этим адресом. В отличие от стандарта ENUM, DUNDi не имеет централизованного управления. Файл dundi.conf содержит добавочные номера DUNDi, используемые для управления тем, какая информация предоставляется; также в нем указаны равноправные участники сети, которым вы предоставляете и от которых будете принимать запросы поиска. Протокол DUNDi рассматривался в главе 14.

## enum.conf

Система электронной нумерации (Electronic Numbering, ENUM) в сочетании с интернет-системой DNS используется для преобразования стандартных номеров E.164 ITU (обычных телефонных) в адреса электронной почты и веб-сайтов, VoIP-адреса и т. п. В DNS ENUM-номер создается путем расположения цифр телефонного номера в обратном порядке, разделения их точками и добавления в начало e164.arpa (основная DNS-зона). Если требуется, чтобы Asterisk выполняла поиск ENUM, необходимо в файле enum.conf сконфигурировать домены, в которых должен осуществляться поиск. Кроме официального домена e164.arpa, можно настроить Asterisk на выполнение поиска в открытом доступном домене e164.org.

## extconfig.conf

Asterisk может записывать конфигурационные данные в базу данных и загружать их оттуда, используя внешний механизм конфигурации (также известный как *архитектура реального времени*). Он обеспечивает интеграцию между внешними конфигурационными файлами (статические отображения) и базой данных, позволяя извлекать информацию из базы данных. Также с ним вы можете проецировать специальные записи реального времени, благодаря чему можно динамически создавать и загружать объекты, сущности, равноправных участников и т. д. без перезагрузки системы. Эти отображения назначаются и конфигурируются в файле extconfig.conf, который используется и res_odbc, и архитектурой реального времени.

## extensions.conf

Диалплан – основа всего. Файл extensions.conf – это средство, с помощью которого вы указываете Asterisk, как должны обрабатываться вызовы. Диалплан содержит список инструкций, которые, в отличие от таковых в традиционных системах телефонной связи, являются полностью настраиваемыми. Диалплан набора имеет настолько большое значение, что мы посвятили ему главы 5 и 6, а также приложение B. Так что вперед, читайте и наслаждайтесь!

## extensions.ael

Этот файл аналогичен extensions.conf, но только предназначается для диалпланов, написанных на языке AEL. Когда Asterisk загружает диалплан, она читает диалплан на AEL из файла extensions.ael и объединяет его с диалпланом из extensions.conf.

## features.conf

features.conf, файл, формально известный как parking.conf, содержит конфигурационную информацию, касающуюся парковки и переадресации вызовов. К опциям настройки парковки вызовов относятся:

- Добавочный номер, на который выполняется звонок для парковки вызовов (`parkext =>`).
- Диапазон добавочных номеров, на которые будут парковаться вызовы (`parkpos =>`).
- Контекст, в котором обрабатывается парковка вызовов (`context =>`).
- Как долго вызов может оставаться припаркованным, прежде чем будет выполнен звонок на добавочный номер, припарковавший его (`parkingtime =>`).
- Звуковой файл, воспроизводимый вызывающему абоненту при снятии с парковки его ранее припаркованного вызова (`courtesytone =>`).
- Объявления ADSI для парковки (`asdipark=yes|no`).

Кроме опций парковки вызовов, в этом файле можно назначить кнопки для слепых переадресаций, переадресаций вручную, записи в одно касание, разъединений, а также добавочный номер перехвата вызовов (который позволяет отвечать на вызов добавочного номера удаленно).

## festival.conf

Механизм речевого воспроизведения текста Festival позволяет Asterisk читать текстовые файлы конечному пользователю генерируемым компьютером голосом. Festival рассматривается в главе 14.

## followme.conf

Термин «найди меня/следуй за мной» создает впечатление наличия в системе офисной АТС настолько развитой логики, что позволяет ей находить пользователей, где бы они ни были, как будто звонки к ним «находят их и следуют за ними». Файл followme.conf используется для конфигурации приложения диалплана `FollowMe()`.

## func_odbc.conf

Функция диалплана `func_odbc` была одной из наиболее ожидаемых возможностей, добавленных в Asterisk 1.4. Эта функция обеспечивает простой механизм соединения с базами данных ODBC посредством диалплана. SQL-запросы описываются в этом конфигурационном файле, а функция диалплана создается автоматически.

## gtalk.conf

В этом конфигурационном файле определяются параметры для соединения с Google Talk.

## http.conf

В Asterisk встроен очень простой демон HTTP, который используется Asterisk GUI и AJAM. Эта функциональность обсуждается в главе 11.

## iax.conf

Аналогично sip.conf, в файле iax.conf описываются опции, касающиеся протокола IAX. Также в нем конфигурируются ваши конечные устройства и провайдеры сервисов. Файл iax.conf подробно рассматривается в приложении A.

## iaxprov.conf

Благодаря этому файлу Asterisk может подготавливать к работе и обновлять встроенные программы устройства IAXy.

## indications.conf

Файл indications.conf используется, чтобы указать Asterisk, как генерировать различные звуки системы телефонной связи, характерные для разных частей мира. Например, английский тональный сигнал готовности линии очень отличается от тонального сигнала в Канаде, но

ваша система Asterisk с готовностью предоставит вам те звуки, которые вы хотите услышать. Этот файл состоит из списка звуков, воспроизводимых системой телефонной связи (сигнал готовности линии, сигналы «занято» и т. д.), с указанием частот, используемых для генерирования этих звуков.

По умолчанию (и без файла indications.conf) Asterisk будет использовать тоны, принятые в Серверной Америке. Можно изменить страну по умолчанию для своей системы, задав двухбуквенный код страны в разделе `[general]`. Список поддерживаемых кодов стран представлен в файле indications.conf.sample, который находится в папке /usr/src/asterisk/configs. Если вы располагаете необходимой информацией, то сможете без труда добавить свою страну. Вот как выглядит конфигурация для Северной Америки:

```
[general]
country=us
;
[us]
description = United States / North America
ringcadance = 2000,4000
dial = 350+440
busy = 480+620/500,0/500
ring = 440+480/2000,0/4000
congestion = 480+620/250,0/250
callwaiting = 440/300,0/10000
dialrecall = !350+440/100,!0/100,!350+440/100,!0/100,!350+440/100,!0/100,350
+440
record = 1400/500,0/15000
info = !950/330,!1400/330,!1800/330,0
```

# jabber.conf

Файл jabber.conf определяет информацию, необходимую для взаимодействия Asterisk с сервером XMPP (Jabber).

# logger.conf

Файл logger.conf определяет тип и детальность сообщений, записываемых в различные файлы журналов в папке /var/log/asterisk/. В нем есть два раздела: `[general]` и `[logfile]`.

## [general]

Параметры раздела `[general]` используются для настройки вывода журналов (их можно и не задавать, поскольку настроек по умолчанию вполне достаточно в большинстве случаев). Однако, если вы любите настраивать подобные вещи, читайте дальше.

С помощью параметра dateformat можно определить, как именно должны выглядеть временные метки:

```
dateformat=%F %T
```

На оперативной странице руководства для Linux по strftime(3) (man strftime) перечислены все способы сделать это.

Чтобы имена файлов журналов начинались с имени хоста вашей системы, задайте параметр appendhostname=yes. Это может быть полезным, если файлы журнала предоставляются вам несколькими системами.

Если по какой-то причине вы не хотите протоколировать события из своих очередей, можно задать параметр queue_log=no.

Если общие события вас не интересуют, укажите Asterisk не включать их в файлы журналов, задав параметр event_log=no.

## [logfiles]

Раздел [logfiles] определяет, информация какого типа будет протоколироваться. Протоколируется разнообразнейшая информация, поэтому желательно распределить записи журналов в различные файлы. Общий формат строк раздела [logfiles] – *имяфайла* => *уровни*, где *имяфайла* – имя файла для хранения протоколируемой информации, а *уровни* – типы сохраняемой информации.

Применение console в качестве имени файла (*имяфайла*) – специальное исключение, которое позволяет управлять типом информации, передаваемой в консоль Asterisk.

Раздел [logfiles] может выглядеть, например, так:

```
[logfiles]
console => notice,warning,error
messages => notice,warning,error
```

Можно задать протоколирование следующей информации:

debug

Активация отладки обеспечивает намного более детальный вывод о том, что происходит в системе. Например, когда активирована отладка, можно увидеть, какие DTMF-тоны вводят абоненты при доступе к своим ящикам голосовой почты. Информация отладки не должна протоколироваться только при фактической отладке чего-либо, поскольку в этом случае файлы журналов очень быстро достигнут огромных размеров.

verbose

Подключившись к консоли Asterisk и задав уровень детальности 3 и выше, в консоли можно увидеть вывод, информирующий о том, что делает Asterisk. Добавив строку verbose_log => verbose в файл logger.conf, этот вывод можно сохранять в файл журнала. Обратите внимание, что при большой детальности сообщений жесткий диск заполнится очень быстро.

`notice`

*Извещение* используется для информирования о малейших изменениях системы, таких как изменения состояния равноправного участника сети. Эти сообщения – вполне обычное явление, а события, о которых они сообщают, как правило, не оказывают негативного воздействия на сервер.

`warning`

*Предупреждение* формируется, когда Asterisk не удается выполнить какое-то действие. Такие типы ошибок, как правило, не являются фатальными, но требуют рассмотрения, особенно если их много.

`error`

*Ошибки* часто бывают связаны с нехваткой памяти. Обычно это свидетельствует о серьезных проблемах, которые могут привести к сбою в работе или «зависанию» системы Asterisk.

## manager.conf

Интерфейс Asterisk Manager – это API, который может использоваться внешними программами для связи и управления Asterisk во многом так же, как вы делали бы это из консоли Asterisk.

Интерфейс Manager обеспечивает программам возможность выполнять команды и запрашивать информацию с сервера Asterisk. Однако это небезопасно; по умолчанию для аутентификации он использует незашифрованные пароли, и все подключенные терминалы по умолчанию получают все события. Интерфейс Asterisk Manager должен использоваться только в доверяемой локальной сети или локально на сервере. Структуры `permit` и `deny` позволяют ограничить доступ к определенным добавочным номерам или подсетям.

Многие доступные Asterisk графические интерфейсы, такие как Flash Operator Panel, используют Manager для извлечения данных и определения статуса приложения. Файл manager.conf описывает способ аутентификации программ в интерфейсе Manager.

Команды Manager (список которых можно получить, введя команду `show manager commands` в консоли Asterisk) имеют различную степень привилегированности. Правами на чтение и запись этих команд можно управлять с помощью опций `read` и `write` в файле manager.conf.

Вот пример файла manager.conf:

```
[general]
enabled = no
port = 5038
bindaddr = 0.0.0.0
[oreilly]
```

```
secret = notvery
deny=0.0.0.0/0.0.0.0
permit= 192.168.1.0/255.255.255.0
read = system,call,log,verbose,command,agent,user,config
write = system,call,log,verbose,command,agent,user,config
```

Подробнее об интерфейсе Asterisk Manager рассказывается в главе 10.

# meetme.conf

MeetMe – одно из самых замечательных приложений Asterisk. Оно позволяет настраивать предопределенные аудиоконференц-залы. Эта довольно простая концепция оказалась исключительно дорогой для реализации во всех остальных офисных АТС. Но то, что кажется чем-то экстраординарным для них, элементарно для Asterisk. Сегодня, или используя выделенный сервер, или посредством сервиса, Asterisk предоставляет эту функциональность как стандартное приложение.

Конференции MeetMe можно создавать динамически с помощью флага `d` в приложении `Dial()` или статически в файле meetme.conf. Для создания конференц-залов используется следующий формат:

```
conf => номер_конференции[,пин][,пин_администратора]
```

Все конференции должны быть определены под заголовком раздела `[rooms]`.

```
[rooms]
conf => 4569
conf => 5060,54377017
conf => 3389,4242,1337
conf => 333,,2424
```

# mgcp.conf

Протокол контроля медиа-шлюзов MGCP (Media Gateway Control Protocol) имеет лишь примитивную поддержку в Asterisk. Скорее всего, это можно объяснить тем, что SIP затмил все остальные VoIP-протоколы (кроме IAX, конечно). Из-за этого использовать MCGP-канал Asterisk в среде производственной эксплуатации можно, только если вы готовы к всестороннему тестированию, желаете платить за соответствующие версии функций и патчей и имеете собственного эксперта по этому протоколу.

Сказав это, мы не собирались объявить, что MGCP мертв. SIP пока что не стал панацеей, как часто утверждают, и MGCP доказал свою полезность в магистральных средах поставщиков услуг связи. Многие верят, что MGCP заполнит нишу или пустующую область, которая еще не открыта, и мы сохраняем интерес к нему.

## modem.conf

Файл modem.conf используется Asterisk для связи с интерфейсами ISDN-BRI через драйвер ISDN4Linux. Поскольку ISDN4Linux не имеет многих основных функций ISDN, как правило, он не используется. Вероятно, самым популярным дополнением для BRI является chan_capi, которое можно найти по адресу *http://www.junghanns.net*.

## musiconhold.conf

Файл musiconhold.conf используется для конфигурации разных классов музыки, используемых в приложениях музыки во время ожидания, и их местоположений. Asterisk может применять для воспроизведения музыки во время ожидания файлы в любом собственном формате. Asterisk также использует определенную версию mpg123 для воспроизведения MP3-файлов, но это не рекомендуется. Можно задать аргументы для класса, что позволяет использовать внешнее приложение для потоковой передачи музыки как локально, так и по сети.

## osp.conf

Протокол открытого взаимодействия OSP (Open Settlement Protocol) официально описан в ETSI TS 101 321, документе Европейского института стандартов по телекоммуникациям (European Telecommunication Standards Institute, ETSI), являющемся результатом деятельности рабочей группы TIPHON. Похоже, OSP – это еще одна попытка применить старое мышление телекоммуникационной отрасли к нарушающим установившийся порядок технологиям.

## oss.conf

С помощью файла oss.conf Asterisk конфигурируется для использования драйвера OSS (Open Sound System – открытая звуковая система), чтобы сделать возможным обмен информацией со звуковой картой по каналу `CONSOLE`. Обратите внимание, что сегодня предпочтительным интерфейсом для канала `CONSOLE` является ALSA.

## phone.conf

Файл phone.conf используется для конфигурации платы Quicknet PhoneJACK. Плата PhoneJACK предоставляет нечто подобное интерфейсу FXS с той точки зрения, что к ней можно подключить аналоговый телефон и передавать вызовы через Asterisk.

# privacy.conf

Файл privacy.conf используется для управления максимальным числом попыток ввода пользователем его 10-значного телефонного номера в приложении `PrivacyManager()`. Приложение `PrivacyManager()` определяет, задан ли Caller ID (ID звонящего) для входящего вызова. Абонент делает столько попыток ввести свой 10-значный номер, сколько указано в privacy.conf; приложение задает для переменной канала `PRIVACYMGR STATUS` значение `SUCCESS` или `FAILED`. Если Caller ID задан, приложение не выполняет никаких действий.

Приложение `PrivacyManager()` также может принимать аргументы в диалплане. Таким образом, можно оставить значение в памяти, а не использовать операцию ввода/вывода на диск для чтения конфигурационного файла, но это будет иметь смысл только при интенсивном использовании данного приложения (много вызовов в секунду).

# queues.conf

Asterisk предоставляет базовую функциональность центра обработки вызовов через свою систему очередей вызовов. Однако те, кто использовал ее в более ответственных средах, часто сообщают о необходимости дополнительной настройки решений. Эта настройка может быть выполнена в файле queues.conf.

В разделе `[general]` файла queues.conf располагаются настройки, которые будут применяться ко всем очередям вызовов. Если для параметра `persistentmembers` присвоено значение `yes`, участник, добавляемый в систему посредством приложения `AddQueueMember()` или интерфейса Asterisk Manager, будет храниться в AstDB и, следовательно, сохраняться при перезагрузках системы.

Параметр `autofill` (автозаполнение) позволяет Asterisk более эффективно распределять звонки между участниками обработки очереди вызовов, особенно если в очереди находятся несколько вызывающих абонентов и несколько агентов обработки вызовов могут принять звонок. Рекомендуется задавать для `autofill` значение `yes`.

Другой общий параметр queues.conf – `MonitorType` (тип записи разговора). Если задано значение `MixMonitor`, входящий и исходящий звуковые потоки будут записываться вместе. Если задано значение `Monitor`, будет использоваться более старый метод записи входящего и исходящего аудиопотоков в разных файлах.

Далее можно описать одну или более очередей, указывая имя очереди в квадратных скобках (`[]`). Для каждой очереди вызовов доступны следующие параметры:

`musiconhold` (музыка во время ожидания)

Этот параметр позволяет задавать класс музыки во время ожидания (сконфигурированный в файле musiconhold.conf), используемый для очереди.

`announce` (приветствие)

Когда вызов представляется участнику обработки очереди вызовов, этому агенту будет воспроизведено приглашение, заданное параметром `announce`, перед установлением соединения с вызывающим абонентом. Это может быть полезным для агентов, зарегистрированных в нескольких очередях. Можно задать или полный путь к файлу, или путь относительно папки /var/lib/asterisk/sounds/.

`strategy` (стратегия)

Asterisk может использовать шесть стратегий распределения вызовов между агентами:

`ringall` (звонить всем)

Очередь звонит всем доступным агентам и устанавливает соединение с агентом, ответившим первым (это по умолчанию).

`roundrobin` (циклический) – устаревший

Очередь последовательно перебирает всех агентов до тех пор, пока не найдет того, кто может принять вызов. `roundrobin` не учитывает загруженности агентов. Также, поскольку `roundrobin` всегда начинает с первого агента в очереди, эта стратегия подходит только в среде, где агенты более высокого ранга должны обрабатывать все вызовы, и только в случае их занятости принять вызов могут агенты с более низким рангом.

`leastrecent` (самый давний)

Вызов направляется на обработку агенту, который не получал вызовы дольше всех.

`fewestcalls` (меньше всего вызовов)

Вызов направляется на обработку агенту, который получил меньше всего звонков. Эта стратегия не учитывает фактической загруженности агента; она учитывает только количество принятых им вызовов (например, агент, принявший 3 вызова по 10 мин каждый, будет предпочтительнее агента, принявшего 5 вызовов по 2 мин каждый).

`random` (случайный)

Как следует из имени этого параметра, выбор агента осуществляется случайным образом. Для небольшого центра обработки вызовов эта стратегия, вероятно, наиболее предпочтительная.

`rrmemory`

Очередь перебирает список обработчиков очереди вызовов, отслеживая, кто из них получил вызов последним. При поступлении следующего вызова Asterisk начнет с этого участника. (Эта стратегия известна как *циклическая память* (round-robin memory)). Она обеспечивает более или менее равномерное распределение вызовов между агентами.

`servicelevel` (уровень обслуживания)

В центре обработки вызовов уровень обслуживания представляет максимальное время ожидания вызывающего абонента перед тем, как его звонок будет передан на обработку агенту. Например, если для `servicelevel` задано значение 60 и показатель уровня обслуживания равен 80%, это означает, что 80% поступивших в очередь вызовов были переданы агентам в течение менее 60 с.

`context`

Если для очереди задан контекст, вызывающий абонент, нажав всего одну цифру, сможет выйти на соответствующий добавочный номер в заданном контексте, если тот существует. Это действие выводит вызывающего абонента из очереди вызовов, то есть он потеряет свое место в очереди – об этом необходимо помнить при использовании данной функции.

`timeout`

Значение `timeout` определяет максимальную продолжительность попыток дозвониться агенту (в секундах), по истечении которого агент будет признан недоступным, а вызов будет возвращен в очередь.

`retry`

По истечении времени ожидания значение параметра `retry` определяет, сколько секунд необходимо подождать, прежде чем представлять вызов вновь доступному агенту.

`weight` (вес)

Параметр `weight` определяет ранг очереди вызовов. Если вызовы ожидают в нескольких очередях, первыми агентам будут предлагаться очереди с большим значением `weight`. При проектировании очередей вызовов необходимо помнить следующее: эта стратегия может привести к тому, что вызов из очереди с самым низким весом никогда не получит ответа. Всегда необходимо гарантированно обеспечить перевод вызовов из очередей с более низким весом в очереди с более высоким весом, чтобы не допустить бесконечного ожидания ответа.

`maxlen` (максимальная длина)

`maxlen` – это максимальное количество вызовов, которое может быть добавлено в данную очередь, прежде чем вызов перейдет к следующему приоритету текущего добавочного номера.

`announce-frequency` **(частота объявлений)**

Значение `announce-frequency` (задаваемое в секундах) определяет, как часто вызывающему абоненту объявляется его место в очереди вызовов и предполагаемое время ожидания.

`announce-holdtime` **(объявить время ожидания)**

Существует три допустимых значения этого параметра: `yes`, `no` и `once`. Параметр `announce-holdtime` определяет, должно ли быть включено предполагаемое время ожидания в объявление места в очереди вызовов. Если задано значение `once`, оно будет озвучено для вызывающего абонента только один раз.

`monitor-format` **(формат записи)**

Этот параметр принимает три возможных значения: `wav`, `gsm` и `wav49`. Активируя эту опцию, вы сообщаете Asterisk о своем желании записывать все выполненные вызовы в очереди в заданном формате. Если эта опция не задана, звонки не будут записываться.

`monitor-join` **(объединить запись)**

Приложение `Monitor()` в Asterisk обычно выполняет запись каждого из направлений разговора в отдельный файл. Задавая параметру `monitor-join` значение `yes`, мы указываем Asterisk объединять эти файлы в конце разговора. Этот параметр следует задавать, только если задан параметр `MonitorType` для приложения `Monitor`.

`joinempty`

Этот параметр принимает три значения: `yes`, `no` и `strict`. Позволяет определять возможность добавления вызывающих абонентов в очередь вызовов на основании статуса участников обработки очереди. Опция `strict` запретит добавление вызывающих абонентов в очередь, если все участники обработки вызовов недоступны.

`leavewhenempty` **(покинуть, если пустая)**

Этот параметр определяет, будут ли ожидающие ответа вызывающие абоненты удалены из очереди вызовов при возникновении условия, не допускающего добавления абонента в очередь (то есть когда все ваши агенты вышли из системы и ушли домой).

`eventwhencalled` **(событие при поступлении вызова)**

Задайте для `eventwhencalled` значение `yes`, если вы хотите, чтобы события очереди передавались в интерфейс Manager.

`eventmemberstatusoff` **(отключить формирование дополнительной информации)**

Задав для этого параметра значение `no`, можно обеспечить формирование дополнительной информации о каждом участнике обработки очереди вызовов.

`reportholdtime` (сообщить время ожидания)

Если для этого параметра задано значение `yes`, участнику обработки очереди вызовов, ответившему на звонок, будет объявлено, сколько времени вызывающий абонент ожидал соединения.

`memberdelay` (задержка участника)

Этот параметр определяет, будет ли задержка между моментом выявления очередью свободного агента и моментом соединения вызова с этим агентом.

`member => имя_участника`

Участниками обработки очереди могут быть или типы каналов, или агенты. Все агенты, перечисленные здесь, должны быть описаны в файле agents.conf.

## res_odbc.conf

Назначение модуля res_odbc.so – сохранение информации конфигурационного файла в базу данных и извлечение этой информации из базы данных. Файл res_odbc.conf определяет, как организуется доступ к таблице базы данных. Файл extconfig.conf используется для описания способа соединения с базой данных.

## res_snmp.conf

Файл res_snmp.conf используется для конфигурации поддержки протокола SNMP (Simple Network Management Protocol – простой протокол управления сетью) в Asterisk. В разделе `[general]` имеется две опции. Опция `subagent` (субагент) определяет, должен ли res_snmp выполняться как субагент или как полноправный SNMP-агент. В Asterisk по умолчанию он выполняется как субагент. Опция `enabled` (активирован) определяет, активирована ли поддержка SNMP в Asterisk. Значение по умолчанию – `no`, его придется изменить, если необходима поддержка SNMP.

## rpt.conf

Файл rpt.conf используется для конфигурации последнего проекта Джима Диксона, Jim's Radio Repeater Application (Программный ретранслятор Джима) (app_rpt). Он обеспечивает возможность Asterisk обмениваться информацией, используя VoIP с применением технологии ретранслятора. Это позволяет эффективно обеспечивать большую зону покрытия беспроводными сетями и информацией маршрутизации для радиолюбителей через их высокоскоростные локальные интернет-соединения.

# rtp.conf

Файл rtp.conf управляет портами транспортного протокола реального времени RTP (Real-time Transport Protocol), используемым Asterisk для формирования и приема RTP-трафика. Протокол RTP используется SIP, H.323, MGCP и, возможно, другими протоколами для передачи медиа-данных между конечными точками.

По умолчанию файл rtp.conf использует порты RTP в диапазоне от 10000 до 20000. Однако, скорее всего, для работы вам понадобится намного меньше портов и многие сетевые администраторы не захотят создавать такое большое окно в своем межсетевом экране. Ограничить диапазон RTP-портов можно, изменив его верхнюю и нижнюю границы в файле rtp.conf.

Обычно для двустороннего SIP-звонка между двумя конечными точками используется пять портов: порт 5060 для обмена служебными SIP-сигналами, по одному порту для потока данных и для протокола управления передачей в реальном времени RTCP (Real-Time Control Protocol) в одном направлении и еще два порта для потока данных и RTCP в противоположном направлении.

Датаграммы UDP содержат 16-разрядное поле для контроля циклическим избыточным кодом (Cyclic Redundancy Check, CRC), которое используется для проверки целостности заголовка и данных датаграммы. При этом посредством деления полиномов из 64-разрядного заголовка создается 16-разрядная контрольная сумма. Полученное значение помещается в 16-разрядное CRC-поле датаграммы, которое удаленный конец соединения затем может использовать для проверки целостности полученной датаграммы.

Задавая значение `rtpchecksums=no`, мы определяем, что ОС не будет создавать/проверять контрольную сумму UDP для сокетов, используемых RTP. Если добавить эту опцию в файл rtp.conf, он будет выглядеть так:

```
[general]
rtpstart=10000
rtpend=20000
rtpchecksums=no
```

# say.conf

Файл say.conf используется для конфигурации грамматических правил разговорного языка для ряда приложений, таких как `SayNumber()`. Если предполагается использовать в Asterisk язык, не поддерживаемый в настоящее время, можно создать сценарий для его поддержки посредством опций конфигурации в этом файле.

## sip.conf

Файл sip.conf определяет все опции SIP-протокола для Asterisk. Правила аутентификации конечных точек, таких как SIP-телефоны и провайдеры сервисов, также конфигурируются в этом файле. С помощью файла sip.conf Asterisk определяет, какие звонки вы желаете принимать и в какую точку диалплана эти звонки должны направляться. В sip.conf задаются многие связанные с SIP опции, которые подробно рассматривались в приложении A.

## sip_notify.conf

Asterisk может удаленно сообщать SIP-телефону о необходимости перепроверки его конфигурационных файлов или перегружать телефон, передавая особым образом форматированное сообщение `NOTIFY` (определенное в файле sip_notify.conf), особое для каждого производителя. Поскольку эти сообщения для каждого производителя индивидуальные, для различных телефонов реализована разная поддержка.

## skinny.conf

Если вы желаете соединяться с телефонами по узкоспециализированному облегченному протоколу управления клиентом SCCP (Skinny Client Control Protocol) компании Cisco, в файле skinny.conf можно определить параметры и каналы, которые будут использовать этот протокол. Однако, поскольку протокол SCCP является узкоспециализированным, его поддержка в Asterisk далека от идеальной, но неизменно улучшается.

## sla.conf

Даже несмотря на то что Asterisk – современная офисная АТС, многим по-прежнему хочется, чтобы она вела себя как старая коммутируемая система со спаренными линиями. Asterisk может эмулировать малую АТС, конфигурируя спаренные линии в sla.conf.

## smdi.conf

Этот файл конфигурирует интерфейс Station Message Desk Interface. SMDI – очень полезное дополнение Asterisk, поскольку оно позволяет ей выступать в роли системы голосовой почты для устаревших АТС, поддерживающих протокол SMDI.

# udptl.conf

Этот файл используется для конфигурации в Asterisk поддержки пакетов UDPTL (User Datagram Protocol Transport Level – протокол транспортного уровня для передачи UDP-данных через пакетные сети). UDPTL-пакеты – один из транспортов, используемых при передаче факсимильных сообщений по протоколу T.38 по IP-соединениям.

# users.conf

С появлением Asterisk GUI разработчики Asterisk обнаружили, что с его помощью можно создать конфигурационный файл с учетными записями пользователей, а не разбрасывать их по нескольким файлам (таким, как extensions.conf, sip.conf и voicemail.conf). Asterisk GUI также обновляет этот файл при добавлении новых пользователей в систему или при изменении настроек пользователя.

# voicemail.conf

Файл voicemail.conf управляет системой голосовой почты Asterisk (называемой Comedian Mail (Почта комедианта)). Он состоит из трех основных разделов. Первый, `[general]`, определяет общие настройки системы голосовой почты. Второй, `[zonemessages]`, позволяет конфигурировать разные зоны голосовой почты, которые являются совокупностью настроек времени и часовых поясов. В третьем и последнем разделе создаются одна или более групп ящиков голосовой почты, каждая из которых содержит описание почтового ящика. Больше информации о введении возможностей голосовой почты в диалплан можно найти в главе 6.

## Общие настройки голосовой почты

Раздел `[general]` файла voicemail.conf содержит множество разнообразнейших опций, которые оказывают влияние на всю систему голосовой почты:

`format` (формат)

Дает список кодеков, используемых для хранения сообщений голосовой почты. Кодеки должны быть разделены символом вертикальной черты (`|`). Формат, заданный первым, является форматом, используемым при прикреплении сообщения голосовой почты к электронному письму. Значение по умолчанию – `wav49|gsm|wav`. Причина хранения голосовой почты в разных форматах – минимизация объемов перекодировки, выполняемой Asterisk при воспроизведении голосовой почты.

`serveremail` (почтовый сервер)

Указывает адрес электронной почты, с которого должны отправляться уведомления о получении голосовой почты.

`attach` (прикрепить)

Определяет, должна ли Asterisk прикреплять звуковой файл голосовой почты к электронному письму-уведомлению о получении голосовой почты.

`maxmsg`

Устанавливает максимальное число сообщений, которые могут храниться в любой папке голосовой почты.

`maxmessage`

Задает максимальную продолжительность сообщения голосовой почты, в секундах.

`minmessage`

Задает минимальную продолжительность сообщения голосовой почты, в секундах.

`maxgreet`

Задает максимальную продолжительность приветствия голосовой почты, в секундах.

`skipms` (пропустить мс)

Устанавливает, сколько миллисекунд вперед/назад пропустить при нажатии пользователем кнопки перемотки вперед или назад при воспроизведении сообщения.

`maxsilence` (максимальная пауза)

Определяет допустимую продолжительность паузы, в секундах, после которой запись будет завершена.

`silencethreshold`

Устанавливает пороговую продолжительность паузы (что считать «паузой» – чем ниже порог, тем выше чувствительность).

`maxlogins` (максимальное число попыток регистрации)

Задает максимальное разрешенное число неудачных попыток регистрации.

`userscontext` (пользовательский контекст)

Определяет, частью какого контекста голосовой почты, определенного в файле users.conf, должны быть почтовые ящики. Значение по умолчанию – контекст голосовой почты `default`.

`externnotify` (внешнее уведомление)

Предоставляет полный путь и имя файла внешней программы, которая должна выполняться при отправке или доставке голосовой

почты либо при проверке почтового ящика. Может быть задано значение `smdi`, чтобы использовать для внешнего уведомления SMDI. Если задано значение `smdi`, для параметра `smdiport` должен быть задан действительный порт, определенный в файле smdi.conf.

`smdiport` (порт smdi)

Определяет порты связи, используемые SMDI. Значение должно быть действительным портом, заданным в файле smdi.conf. Используется, когда для параметра `externnotify` задано значение `smdi`.

`externpass`

Предоставляет полный путь и имя файла внешней программы, выполняемой при изменении пароля доступа к голосовой почте.

`directoryintro`

Если задан, переопределяет стандартное приветствие при входе в телефонную книгу.

`charset`

Определяет набор символов сообщений голосовой почты.

`adsifdn`

Определяет номер дескриптора функции ADSI для загрузки.

`adsisec`

Устанавливает защитный код блокировки ADSI.

`adsiver`

Указывает номер версии приложения голосовой почты ADSI.

`pbxskip`

Указывает Asterisk не добавлять строку `[PBX]:` в начале строки темы электронного письма-уведомления о получении голосовой почты.

`fromstring`:

Меняет строку `From:` электронных сообщений-уведомлений о получении голосовой почты.

`usedirectory` (использовать телефонную книгу)

Позволяет владельцу почтового ящика выбирать записи из телефонной книги для пересылки и/или создания новых сообщений голосовой почты.

`odbcstorage` (odbc-хранение)

Если Asterisk скомпилирована с поддержкой хранения голосовой почты посредством ODBC, эта опция позволит задать, какое ODBC-соединение использовать. ODBC-соединения описаны в файле res_odbc.conf.

`odbctable` (odbc-таблица)

Эта опция используется в сочетании с `odbcstorage`. Она определяет, какая таблица базы данных будет использоваться для сообщений голосовой почты.

`emailsubject` (тема электронного письма)

Определяет тему электронных сообщений-уведомлений о получении голосовой почты.

`pagerfromstring` (строка от: сообщения на пейджер)

Меняет строку `From:` сообщений-уведомлений о получении голосовой почты, рассылаемых на пейджер.

`emailbody` (тело электронного письма)

Обеспечивает тело электронного письма-уведомления о получении голосовой почты.

Пожалуйста, заметьте, что настройки `emailsubject`, `emailbody`, `pagersubject` и `pagerbody` могут использовать следующие переменные для предоставления более детальной информации о голосовой почте:

- `VM_NAME`
- `VM_DUR`
- `VM_MSGNUM`
- `VM_MAILBOX`
- `VM_CALLERID`
- `VM_CIDNUM`
- `VM_CIDNAME`
- `VM_DATE`

`emaildateformat` (формат даты электронной почты)

Определяет формат даты и времени для исходящих уведомлений, рассылаемых по электронной почте. Для получения более подробной информации по формату обратитесь к оперативной странице руководства по `strftime(3)`.

`mailcmd`

Предоставляет полный путь и имя файла программы Asterisk, используемой для отправки уведомлений по электронной почте. Эта опция полезна, если требуется переопределить программу электронной почты по умолчанию.

`nextaftercmd`

Переходит к следующему сообщению, если пользователь нажал кнопку 7 или 9, чтобы удалить или сохранить текущее сообщение. В настоящее время может быть задано только глобально, не для каждого почтового ящика в отдельности.

## Зоны голосовой почты

Поскольку пользователи голосовой почты могут находиться географически в разных местах, Asterisk предоставляет возможность сконфигурировать часовой пояс и способ объявления времени для разных вызывающих абонентов. Каждая уникальная комбинация называется зоной голосовой почты. Конфигурация зон голосовой почты осуществляется в разделе `[zonemessages]` файла voicemail.conf. После этого можно задать, чтобы ящики голосовой почты использовали настройки одной из этих зон.

Каждое описание зоны голосовой почты состоит из строки со следующим синтаксисом:

*имязоны=часовойпояс | формат_времени*

Аргумент *имязоны* – произвольное имя, используемое для обозначения зоны. Аргумент *часовойпояс* – это имя часового пояса системы, определенного в /usr/share/zoneinfo. Аргумент *формат_времени* определяет, как должно проговариваться время системой голосовой почты. Аргумент *формат_времени* образован следующими элементами:

' *имяфайла* '

Имя звукового файла, который будет воспроизводиться (должно быть обязательно заключено в одинарные кавычки).

${*VAR*}

Подстановка переменной.

A или a

День недели (суббота, воскресенье и т. д.).

B, или b, или h

Название месяца (январь, февраль и т. д.).

d или e

Порядковый номер дня месяца (первое, второе… тридцать первое).

Y

Год.

I или l

Час в 12-часовом формате.

H

Час в 24-часовом формате; одноразрядные значения часов начинаются с нуля.

k

Час в 24-часовом формате; одноразрядные значения часов не начинаются с нуля.

`M`

Минуты.

`P` или `p`

A.M. (до полудня) или P.M. (после полудня).

`Q`

«Сегодня», «вчера» или ABdY (примечание: нестандартное значение `strftime`).

`q`

«» (на сегодня), «вчера», день недели или ABdY (примечание: нестандартное значение `strftime`).

`R`

Время в 24-часовом формате, включая минуты.

Например, следующий фрагмент кода задает две разные зоны голосовой почты: одну – для центрального часового пояса в 12-часовом формате, а вторую – для часового пояса Горных штатов в 24-часовом формате:

```
[zonemessages]
central=America/Chicago|'vm-received' Q 'digits/at' IMp
mountain24=America/Denver|'vm-received' q 'digits/at' H 'digits/hundred' M
'hours'
```

## Определение контекстов и ящиков голосовой почты

Теперь, когда общесистемные настройки и зоны голосовой почты заданы, можно определить контексты голосовой почты и индивидуальные почтовые ящики.

Контексты голосовой почты используются для разделения разных групп пользователей голосовой почты. Например, если в Asterisk хранится голосовая почта нескольких компаний, необходимо поместить почтовые ящики каждой компании в разные контексты голосовой почты, чтобы не смешивать их почту. Также контексты голосовой почты можно использовать для создания телефонных книг отделов.

Чтобы определить новый контекст голосовой почты, просто заключите имя контекста в квадратные скобки:

```
[default]
```

В контексте голосовой почты для описания каждого почтового ящика используется следующий синтаксис:

*почтовыйящик*=*пароль*,*имя*[,*email*[,*email_пейджера*[,*опции*]]]

Аргумент *почтовыйящик* – это номер почтового ящика.

Аргумент *пароль* – это числовой код, который должен ввести владелец почтового ящика для доступа к своей голосовой почте. Если пароль начинается со знака минус (-), владелец почтового ящика не может менять пароль.

Если в качестве пароля задано `d`, предполагается, что данная строка предоставляет альтернативное имя, которое можно использовать для этого почтового ящика в телефонной книге. В следующем примере добавочный номер 123 можно найти по именам Robert или Bob, а также часто употребляемому ошибочному написанию его фамилии:

```
123 => 4444,Robert Schauerhamer
123 => d,Bob Schauerhamer
123 => d,Robert Showerhammer
```

Аргументы *email* и *email_пейджера* – это адреса электронной почты, на которые будут отправляться уведомления о получении голосовой почты. Они могут оставаться пустыми, если нет необходимости рассылать уведомления о поступлении голосовой почты по электронной почте. Сообщение, отправляемое по адресу *email_пейджера*, обычно короче и подходит для отправки на мобильный телефон (по электронной почте на SMS-шлюз) или на буквенно-цифровой пейджер.

Аргумент *опции* – это разделенный символами вертикальной черты список опций голосовой почты, которые могут быть определены для почтового ящика. (Эти опции также могут быть заданы глобально в разделе `[general]`.) К действительным опциям голосовой почты относятся:

`tz` (часовой пояс)

Задает зону голосовой почты, определенную ранее в разделе `[zonemessages]`. Эта опция не имеет значения, если для параметра `envelope` задано значение `no`.

`attach`

Прикрепляет сообщение голосовой почты к электронному письму-уведомлению (но не к сообщению на пейджер). Может принимать значение `yes` или `no`.

`attachfmt`

Определяет формат сообщения голосовой почты, которое должно быть прикреплено к сообщению, отправляемому как уведомление по электронной почте. Обычно Asterisk использует первый формат, заданный в параметре `format` раздела `[general]` (рассматривался ранее), но это можно переопределить для каждого почтового ящика отдельно. Эта опция может быть задана только для почтового ящика отдельно.

Эта опция часто используется, если пользователи Windows хотят получать вложения в формате wav49, а пользователи Linux предпочитают формат gsm.

`saycid`

Информация Caller ID (ID звонящего) проговаривается перед сообщением.

`cidinternalcontexts`

Указывает, чтобы при воспроизведении информации Caller ID в качестве имени предоставлялся не добавочный номер, а внутренний контекст.

`sayduration` (проговаривать продолжительность)

Включает/отключает воспроизведение перед сообщением информации о его длительности. Значение по умолчанию – `on`.

`saydurationm`

Определяет минимальную продолжительность, о которой сообщается, если для опции `sayduration` задано значение `on`. Значение по умолчанию – две минуты.

`dialout` (набор номера из)

Определяет контекст, из которого набирается номер (путем выбора опции 4 расширенного меню). Если не задана, набор номера из системы голосовой почты запрещен.

`sendvoicemail` (отправка голосовой почты)

Определяет контекст, из которого будет отправляться голосовая почта (путем выбора опции 5 расширенного меню). Если не задана, отправка сообщений из системы голосовой почты запрещена.

`searchcontexts` (контексты поиска)

По умолчанию Asterisk выполняет поиск только в контексте `default`, если не задан другой контекст. Чтобы Asterisk выполняла поиск во всех контекстах, задайте для этой опции значение `yes`.

`callback` (обратный вызов)

Определяет контекст, из которого будет осуществляться обратный вызов. Если не задан, обратный вызов отправителя из системы голосовой почты запрещен.

`review` (просмотр)

Позволяет отправителю просматривать/перезаписывать свои сообщения перед сохранением. Значение по умолчанию – `off`.

`operator` (оператор)

Позволяет отправителю нажимать кнопку 0 для доступа к оператору перед, после или во время передачи сообщения голосовой почты. Значение по умолчанию – `off`.

`envelope` (конверт)

Включает/выключает воспроизведение конверта перед воспроизведением сообщения. Значение по умолчанию – `on`. Не влияет на опцию 3,3 расширенного меню опций.

`delete` (удалить)

Удаляет сообщения голосовой почты с сервера после отправки уведомления. Эта опция может быть задана только для отдельного почтового ящика; предназначена для пользователей, которые хотят получать сообщения голосовой почты только по электронной почте.

`volgain` (увеличить громкость)

Если сообщения голосовой почты, прикрепленные к письмам электронной почты, слишком тихие, можно задать эту опцию, чтобы увеличить громкость сообщения перед его прикреплением к электронному письму-уведомлению.

Эта опция работает, только если в системе Asterisk установлено приложение `sox`.

`forcename` (заставить указать имя)

Вынуждает новых пользователей записывать их имена. Новый пользователь определяется по паролю, повторяющему номер его почтового ящика. Значение по умолчанию – `no`.

`forcegreetings` (заставить записать приветствия)

Вынуждает новых пользователей записывать приветствия. Новый пользователь определяется по паролю, повторяющему номер его почтового ящика. Значение по умолчанию – `no`.

`hidefromdir` (скрыть от телефонной книги)

Скрывать почтовый ящик от телефонной книги. Значение по умолчанию – `no`.

`tempgreetwarn`

Предупреждать пользователей о том, что их временные приветствия до сих пор выключены.

Можно задавать несколько опций, разделяя их символом вертикальной черты, как показано в описаниях почтовых ящиков 102 и 103 ниже.

Вот примеры описаний почтовых ящиков:

```
[default]
; обычный почтовый ящик с уведомлением
; по электронной почте
101 => 4242,Example Mailbox,somebody@asteriskdocs.org

; улучшенный почтовый ящик с уведомлением по электронной
; почте и пейджеру и парой специальных опций
102 => 9855,Another User,another@asteriskdocs.org,pager@asteriskdocs.
org,attach=no|tz=central
```

```
; почтовый ящик без уведомления по электронной почте
; и массой дополнительных опций
103 => 6522,John Q. Public,,,tz=central|attach=yes|saycid=yes|dialout=fromvm
|callback=fromvm|review=yes
```

## vpb.conf

Этот файл используется для конфигурации плат Voicetronix в Asterisk. Больше информации можно найти в файле-шаблоне vpb.conf.

## zapata.conf

Файл zapata.conf используется для описания отношений между Asterisk и драйвером Zaptel. Поскольку файл zapata.conf используется только Asterisk, он располагается вместе с остальными конфигурационными файлами Asterisk в папке /etc/asterisk/. Как и zaptel.conf, файл zapata.conf содержит массу вариантов выбора, отражающих огромное количество разнообразного оборудования, которое он поддерживает, и мы даже не будем пытаться перечислить здесь все эти опции. В данной книге (в главе 3) рассматривались только аналоговые интерфейсы для драйвера Zaptel.

## zaptel.conf

Файл zaptel.conf располагается отдельно от всех остальных CONF-файлов Asterisk; драйвер Zaptel доступен любому приложению, которое может использовать его, поэтому имеет смысл хранить его в другой папке, не в той, которая используется только Asterisk (/etc/). Программа ztcfg проводит синтаксический разбор для конфигурации аппаратных элементов TDM в вашей системе. Вы настраиваете в файле zaptel.conf три основных элемента:

- Способ идентификации интерфейсов платы в плане набора.
- Тип необходимой интерфейсу системы обмена сигналами.
- Тоновый язык, связанный с конкретным интерфейсом, как задано в zonedata.c.

Будьте очень осторожны и случайно не включите свой модуль FXS в телефонную линию. Напряжение телефонной линии, особенно при поступлении вызова, слишком велико для модуля и может нанести ему непоправимый вред, сделав его абсолютно бесполезным!

В файле zaptel.conf мы задаем тип обмена сигналами, который будет использоваться каналом. Также определяется, какие каналы необходимо загрузить. Опции конфигурационного файла – это информация,

которая будет использоваться для конфигурации каналов с помощью команды `ztcfg`. В файле zaptel.conf предлагается так много параметров, потому что механизм телефонии Zaptel используется множеством разнообразнейших интерфейсов PSTN. Также, поскольку данная технология быстро эволюционирует, все, что мы пишем сейчас, может измениться к тому моменту, когда вы это читаете. Поэтому мы не пытаемся перечислить здесь все эти опции.

В данной книге мы сосредоточились на аналоговых интерфейсах Zaptel, предоставляемых платой TDM400P производства компании Digium (см. главу 3).

# E

# Функции диалплана Asterisk

Функции диалплана очень мощны. Начав их использовать, вы уже не сможете себе представить, как обходились без них до этого. Функции используются в диалплане так же, как и переменные. Пожалуй, их можно рассматривать как переменные с развитой логикой (или, для тех, кто пришел из мира баз данных, как переменные с триггерами). При вызове они выполняют определенное действие, и результат становится частью команды, в которую включена эта функция (точно так же, как это было бы с переменной).

## AGENT

***Возвращает информацию об агенте***

AGENT(*idагента*[:*элемент*])

Эта функция позволяет извлекать информацию, касающуюся агентов, и может только возвращать значение, не принимать.

Действительные значения поля *элемент*:

status (*значение по умолчанию*)

Статус агента (LOGGEDIN | LOGGEDOUT).

password

Пароль агента.

name

Имя агента.

mohclass

Класс музыки во время ожидания.

`exten`

Добавочный номер обратного вызова для агента (`AgentCallbackLogin`).

`channel`

Имя активного канала для агента (`AgentLogin`).

## ARRAY

***Позволяет задавать несколько переменных за один раз***

`ARRAY(`*переменная1*`[|`*переменная2*`[...][|`*переменнаяN*`]])`

Список значений, разделенных запятыми, соответствующий этой функции, будет интерпретирован как набор значений, которые должны быть присвоены переменным, чьи имена заданы аргументами этой функции. Функция может только принимать значение, не возвращать.

```
; Задаем var1 значение 1 и var2 значение 2.
exten => 123,1,Set(ARRAY(var1,var2)=1\,2)
```

Не забудьте или экранировать запятые обратным слэшем в файле extensions.conf, или заключить весь аргумент в кавычки, поскольку функция `Set()` может принимать несколько аргументов.

### Смотрите также

`Set()`

## BASE64_DECODE

***Декодирует строку, кодированную в BASE64***

`BASE64_DECODE(`*base64_строка*`)`

Декодирует BASE64-строку. Эта функция только возвращает значение, не принимает.

### Смотрите также

`BASE64_ENCODE()`

## BASE64_ENCODE

***Кодирует строку в формат BASE64***

`BASE64_ENCODE(`*строка*`)`

Кодирует строку в формат BASE64. Эта функция только возвращает значение, не принимает.

### Смотрите также

`BASE64_DECODE()`

## BLACKLIST

***Проверяет наличие CallerID в черном списке***

BLACKLIST() проверяет семейство blacklist базы данных AstDB на наличие в нем указанного идентификатора вызывающего абонента. Возвращает значение 1 или 0.

Эта функция только возвращает значение, не принимает.

### Смотрите также

DB()

## CALLERID

***Возвращает или устанавливает данные Caller ID для канала***

CALLERID(*типданных*[,*необязательный-CID*])

CALLERID() проводит синтаксический разбор строки Caller ID текущего канала и возвращает ее всю или часть, как задано параметром *типданных*. Допустимые типы данных: all, name, num, ani, dnid или rdnis. Может быть задан (необязательно) альтернативный Caller ID, если вы желаете проводить синтаксический разбор этой строки, а не заданного для канала Caller ID.

Функция может как возвращать, так и принимать значение.

## CDR

***Возвращает или устанавливает информацию CDR для данного вызова (которая будет записана в журнал CDR)***

CDR(*имяполя*[,*опции*])

Вот список имен доступных полей CDR:

clid

Доступно только для чтения. Это значение задается функцией CALLERID(all).

lastapp

Доступно только для чтения. Показывает приложение, которое было выполнено последним.

lastdata

Доступно только для чтения. Показывает аргументы, переданные в приложение, которое было выполнено последним.

src

Доступно только для чтения. Это значение задается функцией CALLERID(ani).

dst

Доступно только для чтения. Соответствует последнему добавочному номеру диалплана.

`dcontext`

Доступно только для чтения. Соответствует последнему контексту диалплана.

`channel`

Доступно только для чтения. Имя канала, от которого поступил звонок.

`dstchannel`

Доступно только для чтения. Имя канала, в который поступил звонок.

`disposition` (состояние)

Доступно только для чтения. Максимальное состояние канала. Если задана опция `u`, это значение будет возвращено как целое число, а не строка: `1 = NO ANSWER`, `2 = BUSY`, `3 = FAILED`, `4 = ANSWERED`.

`amaflags`

Доступно для чтения/записи. Флаги, используемые для учета вызовов и времени разговора абонента. Если задана опция `u`, это значение будет возвращено как целое число, а не строка: `1 = OMIT`, `2 = BILLING`, `3 = DOCUMENTATION`.

`accountcode`

Доступно для чтения/записи. Расчетный счет (максимум 19 символов).

`userfield`

Доступно для чтения/записи. Определяемое пользователем поле.

`start`

Доступно только для чтения. Время начала звонка. Если задана опция `u`, это значение будет возвращено как целое число (количество секунд с начала отсчета времени), а не форматированная строка дата/время.

`answer`

Доступно только для чтения. Время установления соединения (поле может быть пустым, если на вызов еще не поступил ответ). Если задана опция `u`, это значение будет возвращено как целое число (количество секунд с начала отсчета времени), а не форматированная строка дата/время.

`end`

Доступно только для чтения. Время завершения звонка (поле может быть пустым, если звонок еще не завершен). Если задана опция `u`, это значение будет возвращено как целое число (количество секунд с начала отсчета времени), а не форматированная строка дата/время.

`duration`

Доступно только для чтения. Разность между временем начала и окончания, в секундах. Может быть равным 0, если звонок еще не завершен.

`billsec`

Доступно только для чтения. Разность между временем установления соединения и окончания звонка, в секундах. Может быть равным 0, если звонок еще не завершен.

`uniqueid`

Доступно только для чтения. Строка, которая будет уникальна для каждого звонка в рамках этого экземпляра Asterisk.

Могут быть заданы следующие опции:

`l`

В случае если используется множество CDR посредством `ForkCDR()`, все результаты для вызова будут извлечены из последней записи параметров вызова.

`r`

Специальные переменные CDR будут извлечены из последней записи параметров вызова, но стандартные поля будут получены из первой записи.

`u`

Будет возвращено значение без синтаксического разбора. Записи, на которые этот флаг оказывает влияние, представлены в списке имен полей выше.

Также можно задать значение *имяполя*, не представленное в приведенном выше списке, и создать собственную переменную, значение которой можно изменять с помощью этой функции. И эта переменная будет храниться в CDR.

### Смотрите также

`CHANNEL()`

## CHANNEL

***Получение или задание различных параметров канала***

`CHANNEL(`*элемент*`)`

Стандартные элементы (для всех типов каналов):

`audioreadformat`

Доступно только для чтения. Формат, используемый для приема аудиоданных в настоящий момент.

`audionativeformat`

Доступно только для чтения. Формат, обычно используемый для аудиоданных.

`videonativeformat`

Доступно только для чтения. Формат, обычно используемый для видеоданных.

`audiowriteformat`

Доступно только для чтения. Формат, используемый для передачи данных в настоящий момент.

`callgroup`

Чтение/запись. Группы вызовов для перехвата входящих вызовов.

`channeltype`

Доступно только для чтения. Технология, используемая для канала.

`language`

Чтение/запись. Язык, на котором записываются и воспроизводятся звуковые файлы.

`musicclass`

Чтение/запись. Класс (из файла musiconhold.conf) музыки во время ожидания.

`rxgain`

Чтение/запись. Уровень принимаемого сигнала (в децибелах) для драйверов каналов, которые это поддерживают.

`txgain`

Чтение/запись. Уровень передаваемого сигнала (в децибелах) для драйверов каналов, которые это поддерживают.

`tonezone`

Чтение/запись. Часовой пояс, соответственно которому генерируются различные сигналы.

`state`

Доступно только для чтения. Текущее состояние канала.

`transfercapability`

Чтение/запись. Что может передаваться по сети ISDN. Текущие действительные значения:

`SPEECH`

Речь (по умолчанию, голосовые звонки).

DIGITAL

Неограниченная цифровая информация (вызовы для передачи данных).

RESTRICTED_DIGITAL

Ограниченная цифровая информация.

3K1AUDIO

Аудиосигнал частотой 3,1кГц (вызовы для передачи факса).

DIGITAL_W_TONES

Неограниченная цифровая информация с тонами/приветствиями.

VIDEO

Видео.

Могут быть доступны дополнительные значения параметра *элемент* в зависимости от драйвера канала; более детальная информация представлена в его документации. В результате любого запроса на элемент, недоступный для текущего канала, будет возвращена пустая строка.

### Смотрите также

CDR()

## CHECK_MD5

***Проверяет хеш MD5***

CHECK_MD5(*хеш*,*данные*)

В случае успешной проверки возвращает 1, иначе 0.

Эта функция признана устаревшей и была заменена функцией MD5() со встроенным синтаксическим анализатором выражений.

### Смотрите также

MD5()

## CHECKSIPDOMAIN

***Проверяет, является ли домен локальным***

CHECKSIPDOMAIN(*домен*|*IP*)

Эта функция проверяет, является ли заданный в аргументе домен локальным SIP-доменом, для обработки которого сконфигурирован данный сервер Asterisk. Возвращает имя домена, если он обрабатывается локально, в противном случае возвращается пустая строка. Обратите внимание на конфигурационную опцию domain в файле sip.conf.

# CURL

***Возвращает данные, полученные в результате операции GET или POST по заданному URL***

CURL(*url*[|*отправляемые-данные*])

По умолчанию CURL() будет выполнять операцию HTTP GET для получения значения *url*. Однако, если задан параметр *отправляемые-данные*, будет выполняться HTTP POST.

## Смотрите также

SendURL()

# CUT

***Вырезает строку на основании заданного разделителя***

CUT(*имяпеременной*, *символ-разделитель*, *диапазон*)

Функция CUT() аналогична инструменту командной строки UNIX cut(1) и, кстати, разработана на базе этого инструмента.

В диалплане можно задавать смещения в символах для выбора подстроки из переменной исключительно на основании постоянной длины символов (а именно 1). Функция CUT() создана, чтобы помочь при работе с данными, которые могут иметь несколько секций переменной длины с одинаковым разделителем.

Самый распространенный вариант – имя канала, которое состоит из двух частей: базового имени и уникального идентификатора (например, SIP/tom-abcd1234 или SIP/bert-1a2b3c4d). Функция CUT() может использоваться для обрезки уникального идентификатора независимо от длины базового имени:

```
; Вырезаем уникальный идентификатор
; из имени текущего канала
exten => 123,1,Set(chan=${CUT(CHANNEL,-,1)})
```

*имяпеременной* – это имя обрабатываемой переменной. Обратите внимание, что функция CUT() работает с именем переменной, а не ее значением. CUT() уникальна с этой точки зрения.

*символ-разделитель* – символ, который будет выступать в роли разделителя (по умолчанию '-').

*диапазон* позволяет определить, какие поля будут возвращены. В качестве параметра *диапазон* с помощью символа - может быть задан диапазон (например, 1-3), а с помощью символа & – группа диапазонов и номеров полей (например, 1&3-4). Заметьте, что, если задано несколько номеров полей, поля результирующего значения будут иметь те же разделители.

Параметр *диапазон* использует смещение, начиная с 1. То есть первое поле – это поле номер 1 (в противоположность смещению с 0, при котором первое поле шло бы под номером 0).

### Смотрите также

FIELDQTY()

## DB

***Выполняет чтение или запись в AstDB***

DB(*семейство/ключ*)

Будет возвращать значение записи базы данных (или пустую строку, если записи не существует) или записывать значение в базу данных.

### Смотрите также

DBdel(), DB_DELETE(), DBdeltree(), DB_EXISTS()

## DB_DELETE

***Удаляет ключ или семейство ключей из базы данных AstDB***

DB_DELETE(*семейство/ключ*)

Возвращает значение из базы данных и удаляет его.

### Смотрите также

DBdel(), DB(), DBdeltree()

## DB_EXISTS

***Проверяет, существует ли в AstDB указанный ключ***

DB_EXISTS(*семейство/ключ*)

Проверяет, существует ли ключ в базе данных Asterisk.

### Смотрите также

DB()

## DUNDILOOKUP

***Запрашивает у равноправных участников системы DUNDi определенный номер***

DUNDILOOKUP(*номер*[|*контекст*[|*опция*)

Выполняет поиск телефонного номера в системе DUNDi.

## ENUMLOOKUP

***Запрашивает в базе данных ENUM определенный номер***

ENUMLOOKUP(*номер*[|*Тип-метода*[|*опции*[|*запись*#[|*суффикс-зоны*]]]])

Позволяет получать основные или специальные NAPTR-записи или счетчики типов NAPTR для ENUM или ENUM-подобных DNS-указателей.

## ENV

***Работает с переменными окружения***

ENV(*имяпеременнойокружения*)

Возвращает или устанавливает значение переменной окружения, заданной параметром *имяпеременнойокружения*.

## EVAL

***Вычисляет хранящиеся переменные***

EVAL(*переменная*)

EVAL() – одна из наиболее мощных функций диалплана. Она позволяет сохранять переменные выражения не только в файле extensions.conf, а, например, и в базе данных и при этом вычислять их в диалплане так же, как если бы они были просто включены в него. Можно смело утверждать, что EVAL() – основной элемент в обеспечении истинной динамичности диалплана.

```
; Мы можем сохранить в записи базы данных для
; extension/123, например, следующее:
; "SIP/${DB(ext2chan/123)}". Это направит нас к другой
; записи базы данных.
exten => _XXX,1,Set(dialline=${DB(extension/${EXTEN})})
exten => _XXX,n,Dial(${EVAL(${dialline})})

; Реальный пример (взят из кода, используемого
; в производственной эксплуатации)
exten => _1NXXNXXXXXX,n(generic),Set(provider=${DB(rt2provider/${route})}-nanp)
exten => _1NXXNXXXXXX,n(provider),Dial(${EVAL(${DB(provider/${provider})})})
exten => _1NXXNXXXXXX,n,Goto(nextroute)
```

### Смотрите также

Exec()

## EXISTS

***Проводит проверку, является ли значение не пустым***

EXISTS(*данные*)

Тест на существование значения: возвращает 1, если оно не пустое; 0 в противном случае.

## FIELDQTY

***Выполняет подсчет полей***

FIELDQTY(*имяпеременной|разделитель*)

Подсчитывает поля с использованием произвольно заданного разделителя.

### Смотрите также

CUT()

## FILTER

***Удаляет из строки запрещенные символы***

FILTER(*разрешеные-символы|строка*)

Выполняет фильтрацию содержимого параметра *строка* и включает в результат только значение *разрешенные-символы*:

```
; Гарантируем, что номер Caller*ID содержит только цифры
exten => Set(CALLERID(num)=${FILTER(0123456789,${CALLERID(num)})})
```

Эта функция только возвращает значение, не принимает.

### Смотрите также

QUOTE()

## GLOBAL

***Указывает глобальное пространство имен***

GLOBAL(*имяпеременной*)

Возвращает или устанавливает значение заданной глобальной переменной.

## GROUP

***Включает канал в заданную группу***

GROUP([*категория*])

Возвращает или устанавливает группу каналов.

```
; Разрешаем одновременно выполнять доступ к системе
; оповещения только одному пользователю.
exten => 8000,1,Set(GROUP()=pager)
exten => 8000,n,GotoIf($[${GROUP_COUNT(pager)} > 1]?hangup)
exten => 8000,n,Page(SIP/101&SIP/102&SIP/103&SIP/104)
exten => 8000,n(hangup),Hangup
```

### Смотрите также

GROUP_COUNT(), GROUP_LIST(), GROUP_MATCH_COUNT()

## GROUP_COUNT

***Подсчитывает количество каналов в заданной группе***

GROUP_COUNT([*имягруппы*][@*категория*])

Подсчитывает количество каналов в заданной группе. Возвратит количество каналов в группе текущего канала, если значение имягруппы не задано.

### Смотрите также

GROUP(), GROUP_LIST(), GROUP_MATCH_COUNT()

## GROUP_LIST

***Возвращает список групп каналов***

GROUP_LIST()([*имягруппы*][@*категория*])

Возвращает список групп, заданных для канала.

### Смотрите также

GROUP(), GROUP_COUNT(), GROUP_MATCH_COUNT()

## GROUP_MATCH_COUNT

***Подсчитывает количество каналов в группах, имена которых соответствуют заданному шаблону***

GROUP_MATCH_COUNT(*шаблониименигруппы* [@*категория*])

Подсчитывает количество каналов в группах, соответствующих заданному шаблону.

### Смотрите также

GROUP(), GROUP_COUNT(), GROUP_LIST()

## IAXPEER

***Получает информацию о IAX-канале***

IAXPEER(*имяравноправногоучастника*[|*элемент*])
IAXPEER(CURRENTCHANNEL[|*элемент*])

Получает информацию о равноправном участнике IAX.

Если имя равноправного участника задано, действительными значениями параметра *элемент* являются:

ip

IP-адрес этого равноправного участника. Если параметр *элемент* не задан, IP-адрес будет предоставлен.

status

Статус равноправного участника (если qualify=yes).

`mailbox`

Заданный для равноправного участника почтовый ящик.

`context`

Заданный для равноправного участника контекст.

`expire`

Время следующего истечения регистрации этого равноправного участника.

`dynamic`

Этот равноправный участник зарегистрирован в Asterisk? (`yes`/`no`)

`callerid_name`

Заданное для этого равноправного участника имя Caller ID.

`callerid_num`

Заданный для этого равноправного участника номер Caller ID.

`codecs`

Заданные для этого равноправного участника кодеки.

`codec[`*x*`]`

Индекс *x* предпочтительного кодека (начиная с нуля).

### Смотрите также

`SIPPEER()`

## IF

***Выбор значения по условию***

`IF(`*выражение*`?[`*true*`][:`*false*`])`

Условный оператор: возвращает данные, следующие за символом `?`, в случае истинности условия, в противном случае возвращает данные, следующие за символом `:`.

```
; Возвращается foo
exten => 123,1,Set(something=${IF($[2 > 1]?foo:bar)})
; Возвращается bar
exten => 123,n,Set(something=${IF($[2 < 1]?foo:bar)})
```

### Смотрите также

`GotoIf()`

## IFTIME

***Сравнивает текущее системное время с заданным***

`IFTIME(`*время*`,`*дни_недели*`,`*дни_месяца*`,`*месяцы*`?[`*true*`][:`*false*`])`

Условный оператор: возвращает данные, следующие за символом `?`, в случае истинности условия, в противном случае возвращает данные, следующие за символом `:`.

*время*

Диапазоны времени в 24-часовом формате.

*дни_недели*

Дни недели (mon, tue, wed, thu, fri, sat, sun).

*дни_месяца*

Дни месяца (1–31).

*месяцы*

Месяцы (jan, feb, mar, apr и т. д.).

### Смотрите также

GotoIfTime()

## ISNULL

***Проверяет, является ли значение пустым***

ISNULL(*данные*)

Возвращает 1, если значение *данные* пустое, или 0 в противном случае.

### Смотрите также

LEN(), EXISTS()

## KEYPADHASH

***Выполняет преобразование букв в числа***

KEYPADHASH(*строка*)

Преобразует буквы в параметре *строка* в эквивалентные цифры номеронабирателя.

```
; Вычислим хеши фамилий авторов. Итак,
; соответствующими значениями будут
; 623736, 76484 и 82663443536.
exten => 123,1,Set(lastname1=${KEYPADHASH(Madsen)})
exten => 123,n,Set(lastname2=${KEYPADHASH(Smith)})
exten => 123,n,Set(lastname3=${KEYPADHASH(VanMeggelen)})
```

### Смотрите также

Directory()

## LANGUAGE

***Определяет язык канала***

LANGUAGE()

Возвращает или устанавливает язык канала.

Эта функция признана устаревшей и была заменена функцией CHANNEL(*язык*).

### Смотрите также

CHANNEL()

## LEN

***Вычисляет длину строки***

LEN(*строка*)

Возвращает длину значения *строка*.

## MATH

***Выполняет математические вычисления***

MATH(*число1 операция число2*[,*тип_результата*])

Выполняет математические функции.

```
exten => 123,1,Set(value1=${MATH(1+2)})
```

## MD5

***Вычисляет хеш MD5***

MD5(*данные*)

Вычисляет хеш MD5 параметра *данные*.

### Смотрите также

SHA1()

## MUSICCLASS

***Выполняет доступ к настройкам музыки во время ожидания для канала***

MUSICCLASS()

Эта функция была признана устаревшей и заменена функцией CHANNEL(*классмузыки*).

Возвращает или устанавливает класс музыки во время ожидания.

### Смотрите также

CHANNEL()

## QUEUE_MEMBER_COUNT

***Подсчитывает количество участников обработки очереди вызовов***

QUEUE_MEMBER_COUNT(*имяочереди*)

Подсчитывает количество участников, отвечающих на вызовы очереди.

### Смотрите также

QUEUE_MEMBER_LIST()

## QUEUE_MEMBER_LIST

***Возвращает список участников обработки очереди вызовов***

QUEUE_MEMBER_LIST(*имяочереди*)

Возвращает список интерфейсов в очереди.

### Смотрите также

QUEUE_MEMBER_COUNT()

## QUEUE_WAITING_COUNT

***Подсчитывает количество вызовов, ожидающих ответа***

QUEUE_WAITING_COUNT(*имяочереди*)

Подсчитывает количество вызовов, стоящих в очереди и ожидающих ответа.

## QUEUEAGENTCOUNT

QUEUEAGENTCOUNT(*имяочереди*)

Эта функция была признана устаревшей и заменена функцией QUEUE_MEMBER_COUNT().

Подсчитывает количество агентов, отвечающих на вызовы очереди.

### Смотрите также

QUEUE_MEMBER_COUNT(), QUEUE_MEMBER_LIST()

## QUOTE

***Экранирует строку***

QUOTE(*строка*)

Заключает в кавычки заданную строку, по необходимости экранируя кавычки, встречающиеся в данной строке.

### Смотрите также

FILTER()

## RAND

***Возвращает случайное число***

RAND([*минимум*][|*максимум*])

Выбирает случайное число из заданного диапазона.

Функция RAND() случайным образом выбирает целое число, находящееся в диапазоне между значениями *минимум* и *максимум*, включая и их самих, и возвращает его. Если значение *минимум* не задано, по умолчанию оно принимается равным 0. Если значение *максимум* не задано, по умолчанию оно принимается равным константе INT_MAX, которая равна 2147 483 647 на 32-разрядных платформах. Обратите внимание, что на 64-разрядных платформах значение INT_MAX немного больше.

## REALTIME

***Извлекает данные реального времени***

REALTIME(*семейство*|*шаблонполя*[|*значение*[||*разделитель1*[|*разделитель2*]]])

Функции для чтения/записи в режиме реального времени. Приведенный выше синтаксис используется для чтения, приведенный ниже – для записи:

REALTIME(*семейство*|*шаблонполя*|*значение*|*поле*)

## REGEX

***Проводит сравнение на основании регулярного выражения***

REGEX("*регулярное выражение*" *данные*)

Выполняет сопоставления на основании регулярного выражения.

## SET

***Задает значение переменной***

SET(*имяпеременной*=[*значение*])

Функция SET присваивает значение переменной канала. Часто применяется для задания значений, содержащих символ |, поскольку этот символ обычно является разделителем при использовании с приложением Set().

### Смотрите также

Set()

## SHA1

***Вычисляет хеш по алгоритму SHA-1***

SHA1(*данные*)

Вычисляет хеш значения *данные* по алгоритму SHA-1.

### Смотрите также

MD5()

## SIP_HEADER

***Извлекает заголовок SIP***

SIP_HEADER(*имя*[,*номер*])

Возвращает SIP-заголовок.

## SIPCHANINFO

***Извлекает информацию о SIP-канале***

SIPCHANINFO(*элемент*)

Возвращает заданный SIP-параметр текущего канала.

Действительные значения параметра *элемент*:

peerip

IP-адрес данного равноправного участника SIP.

recvip

IP-адрес источника сообщений данного равноправного участника SIP.

from

URI SIP из заголовка From:.

uri

URI SIP из заголовка Contact:.

useragent

Имя SIP-агента пользователя.

peername

Имя равноправного участника SIP.

t38passthrough

1, если T38 предлагается или активирован в данном канале, в противном случае 0.

## SIPPEER

***Извлекает информацию о равноправном участнике SIP***

SIPPEER(*имяравноправногоучастника*[|*элемент*])

Возвращает информацию о равноправном участнике SIP.

Действительные значения параметра *элемент*:

ip

IP-адрес данного равноправного участника. Если значение *элемент* не задано, будет указан IP-адрес.

`mailbox`

Заданный для данного равноправного участника почтовый ящик.

`context`

Заданный для данного равноправного участника контекст.

`expire`

Время следующего истечения регистрации.

`dynamic`

Это устройство зарегистрировано в Asterisk? (`yes/no`)

`callerid_name`

Заданное для этого равноправного участника имя Caller ID.

`callerid_number`

Заданный для этого равноправного участника номер Caller ID.

`status`

Статус данного равноправного участника (если `qualify=yes`).

`regexten`

Зарегистрированный для данного равноправного участника добавочный номер, если задан.

`limit`

Ограничение на число одновременных вызовов для данного равноправного участника.

`curcalls`

Текущее количество вызовов. Доступно, только если для данного равноправного участника установлен предел на число одновременных вызовов.

`language`

Язык по умолчанию для данного равноправного участника.

`accountcode`

Код учетной записи данного равноправного участника.

`useragent`

Имя SIP-агента пользователя.

`codecs`

Заданные для этого равноправного участника кодеки.

`codec[`*x*`]`

Индекс *x* предпочтительного кодека (начиная с нуля).

## Смотрите также

`IAXPEER()`

## SORT

***Выполняет сортировку списка***

SORT(*ключ1*:*значение1*[...][,*ключN*:*значениеN*])

Выполняет сортировку списка ключей/значений, создавая список ключей на основании их значений (сортировка по ключу), которые могут быть любыми действительными числами (с плавающей точкой).

## SPEECH

***Возвращает информацию о результатах работы по распознаванию речи***

SPEECH(*аргумент*)

Возвращает информацию о результатах работы по распознаванию речи.

## SPEECH_ENGINE

***Изменяет свойство механизма обработки речи***

SPEECH_ENGINE(*имя*)=*значение*)

Меняет конкретный атрибут механизма обработки речи.

## SPEECH_GRAMMAR

***Возвращает информацию о грамматике речи***

SPEECH_GRAMMAR(*результат номер*)

Возвращает соответствующую информацию по грамматике результата, если таковая доступна.

## SPEECH_SCORE

***Возвращает параметр достоверности результата распознавания речи***

SPEECH_SCORE(*результат номер*)

Возвращает параметр достоверности результата.

## SPEECH_TEXT

***Возвращает текст, распознанный системой распознавания речи***

SPEECH_TEXT(*результат номер*)

Возвращает текст результата, распознанный системой распознавания речи.

## SPRINTF

***Выполняет форматирование строки***

SPRINTF(*формат*|*аргумент1*[|...*аргументN*])

Форматирует переменную или ряд переменных соответственно формату строки.

Чаще всего функция SPRINTF используется для дополнения числа нулями до определенной длины:

```
; Возвращает 00123
exten => 123,1,Set(padfive=${SPRINTF(%05d,${EXTEN})})
```

Большинство опций форматирования, приведенных на странице руководства по sprintf(3), также реализованы в данной функции диалплана.

### Смотрите также

STRFTIME()

## STAT

***Возвращает атрибуты файловой системы***

STAT(*флаг*,*имяфайла*)

Выполняет проверку заданного файла.

Параметр *флаг* может принимать одно из следующих значений:

e

Возвращает 1, если файл существует; 0 в противном случае.

s

Возвращает размер файла в байтах.

f

Возвращает 1, если по указанному пути находится обычный файл (не папка, символическая ссылка, сокет или устройство), или 0 в противном случае.

d

Возвращает 1, если по указанному пути находится папка (не обычный файл, символическая ссылка, сокет или устройство), или 0 в противном случае.

M

Возвращает время последнего изменения содержимого файла в секундах с начала отсчета времени.

C

Возвращает время последнего изменения индексного дескриптора (inode) файла в секундах с начала отсчета времени.

m

Возвращает режим доступа к файлу (как восьмеричное число).

## STRFTIME

***Форматирует дату и время***

STRFTIME([*началоотсчетавремени*][|[*часовойпояс*][|*формат*]])

Возвращает текущую дату/время в заданном формате.

STRFTIME передает аргументы *началоотсчетавремени* и *формат* прямо в базовый вызов библиотеки strftime(3) языка C, поэтому для получения дополнительной информации обращайтесь к странице руководства. В качестве параметра *часовойпояс* должно использоваться имя папки/файла из папки /usr/share/zoneinfo (например, America/Chicago или America/New_York).

### Смотрите также

STRPTIME()

## STRPTIME

***Преобразует строку в дату и время***

STRPTIME(*датавремя*|*часовойпояс*|*формат*)

Возвращает время в секундах с начала отсчета времени для произвольной строки даты/времени, структурированной в заданном формате.

Назначение этой функции – преобразовать отформатированные дату/время обратно в секунды с начала отсчета времени (с полночи по Гринвичу 1 января 1970 года), чтобы можно было производить с ними какие-то вычисления, или просто преобразовать их в какой-то другой формат даты/времени.

STRPTIME передает строку и формат непосредственно в базовый вызов библиотеки функции strptime(3) языка C, поэтому для получения дополнительной информации обращайтесь к странице руководства. В качестве параметра *часовойпояс* должно использоваться имя папки/файла из папки /usr/share/zoneinfo (например, America/Chicago или America/New_York).

### Смотрите также

STRFTIME()

## TIMEOUT

***Работает со значениями времени ожидания канала***

TIMEOUT(*типвремениожидания*)

Возвращает или устанавливает значения времени ожидания для канала.

Может производить действия со следующими типами времени ожидания:

absolute

Абсолютное максимально допустимое время звонка. Задание значения 0 отключает эту возможность.

digit

Максимально допустимая пауза между нажатиями цифр при вводе добавочного номера пользователем. Если пользователь вводит добавочный номер и время ожидания истекло, считается, что ввод за-

вершен, и начинается обработка добавочного номера. Заметьте, что, если введен действительный добавочный номер, для начала его обработки нет необходимости дожидаться истечения времени ожидания, поэтому обычно по истечении указанного времени ожидания добавочный номер считается недействительным (и управление передается добавочному номеру `i`, или, если его не существует, вызов завершается). Время ожидания по умолчанию – 5 с.

`response`

Максимально допустимый промежуток времени после выполнения ряда приоритетов для канала, в течение которого пользователь может начать набор добавочного номера. Если пользователь не вводит добавочный номер в течение этого промежутка времени, управление передается добавочному номеру `t`, если он существует, в противном случае вызов завершается. Время ожидания по умолчанию – 10 с.

## TXTCIDNAME

***Выполняет DNS-поиск***

`TXTCIDNAME(`*номер*`)`

Выполняет поиск вызывающего абонента с помощью DNS.

## URIDECODE

***Декодирует URI***

`URIDECODE(`*данные*`)`

Декодирует строку из формата, предназначенного для безопасного использования в URI, согласно стандарту RFC 2396.

### Смотрите также

`URIENCODE()`

## URIENCODE

***Кодирует URI***

`URIENCODE(`*данные*`)`

Кодирует строку для ее безопасного использования в URI согласно стандарту RFC 2396.

### Смотрите также

`URIDECODE()`

## VMCOUNT

***Подсчитывает количество сообщений голосовой почты***

`VMCOUNT(`*почтовыйящик*`[@`*контекст*`][|`*папка*`])`

Подсчитывает количество сообщений голосовой почты в заданном почтовом ящике.

# F

# Команды интерфейса Asterisk Manager

В данном приложении представлен список команд, которые можно выполнить с помощью интерфейса Asterisk Manager (AMI). Больше информации об AMI можно найти в главе 14.

## AbsoluteTimeout

***Задает максимальное время ожидания для канала***

Выполняет разъединение канала по прошествии определенного времени.

### Параметры

`Channel`

[обязательный] Имя канала, для которого задается максимальное время ожидания.

`Timeout`

[обязательный] Максимальная продолжительность вызова, в секундах.

`ActionID`

[необязательный] Идентификатор, который может использоваться для опознавания ответа на эту команду.

### Примечания

Asterisk подтвердит установку времени ожидания сообщением `Timeout Set`.

### Привилегии

`call, all`

### Пример

```
Action: AbsoluteTimeout
   Channel: SIP/testphone-10210698
   Timeout: 15
   ActionID: 12345

   Response: Success
   Message: Timeout Set
   ActionID: 12345
```

## AgentCallbackLogin

***Регистрирует агента в системе очереди вызовов в режиме обратного вызова***

Регистрирует указанного агента в системе очереди вызовов Asterisk в режиме обратного вызова. При распределении вызова этот агент будет переведен на заданный добавочный номер.

### Параметры

`Agent`

[обязательный] Идентификатор агента для регистрации в системе, как задано в файле agents.conf.

`Exten`

[обязательный] Добавочный номер, используемый для обратного вызова.

`Context`

[необязательный] Контекст, используемый для обратного вызова.

`AckCall`

[необязательный] Если задано значение `true`, агент должен подтвердить прием вызова (агент нажимает кнопку #) при обратном вызове к нему.

`WrapupTime`

[необязательный] Минимальный промежуток времени после разъединения, перед тем как агент получит новый вызов.

`ActionID`

[необязательный] Идентификатор, который может использоваться для опознавания ответа на эту команду.

### Привилегии

agent, all

### Пример

```
Action: AgentCallbackLogin
Agent: 1001
Exten: 201
Context: Lab
ActionID: 24242424

Response: Success
Message: Agent logged in
ActionID: 24242424

Event: Agentcallbacklogin
Privilege: agent,all
Agent: 1001
Loginchan: 201@Lab
```

### Примечания

Команда AgentCallbackLogin (а также приложение AgentCallbackLogin()) была признана устаревшей. Вместо нее предлагается использовать команду QueueAdd. Более подробную информацию можно найти в файле doc/queues-with-callback-members.txt в папке исходного кода Asterisk.

## AgentLogoff

***Отменяет регистрацию агента***

Отменяет регистрацию указанного агента в системе очереди вызовов.

### Параметры

Agent

[обязательный] Идентификатор агента, регистрация которого должна быть отменена.

Soft

[необязательный] Задайте значение true, чтобы не происходило разъединение существующих вызовов.

ActionID

[необязательный] Идентификатор, который может использоваться для опознавания ответа на эту команду.

### Привилегии

agent, all

### Пример

```
Action: AgentLogoff
Agent: 1001
Soft: true
ActionID: blahblahblah

Response: Success
Message: Agent logged out
ActionID: blahblahblah

Event: Agentcallbacklogoff
Privilege: agent,all
Agent: 1001
Reason: CommandLogoff
Loginchan: 201@Lab
Logintime: 5698
```

## Agents

***Представляет список агентов и их статусов***

Данная команда служит для предоставления списка с информацией обо всех сконфигурированных агентах.

### Привилегии

`agent, all`

### Пример

```
Action: Agents
ActionID: mylistofagents

Response: Success
Message: Agents will follow
ActionID: mylistofagents

Event: Agents
Agent: 1001
Name: Jared Smith
Status: AGENT_IDLE
LoggedInChan: 201@Lab
LoggedInTime: 1173237646
TalkingTo: n/a
ActionID: mylistofagents

Event: Agents
Agent: 1002
Name: Leif Madsen
Status: AGENT_LOGGEDOFF
LoggedInChan: n/a
LoggedInTime: 0
```

```
TalkingTo: n/a
ActionID: mylistofagents

Event: Agents
Agent: 1003
Name: Jim VanMeggelen
Status: AGENT_LOGGEDOFF
LoggedInChan: n/a
LoggedInTime: 0
TalkingTo: n/a
ActionID: mylistofagents

Event: AgentsComplete
ActionID: mylistofagents
```

## ChangeMonitor

***Меняет имя файла для записи разговора по каналу***

Команда `ChangeMonitor` может использоваться для изменения файла, запись которого была начата ранее по команде `Monitor`. Для этого используются следующие параметры.

### Параметры

`Channel`

[обязательный] Используется для задания канала, разговоры по которому будут записываться.

`File`

[обязательный] Новое имя файла, в который будут записываться разговоры по каналу.

`ActionID`

[необязательный] Идентификатор, который может использоваться для опознавания ответа на эту команду.

### Привилегии

`call`, `all`

### Пример

```
Action: ChangeMonitor
Channel: SIP/linksys-084c63c0
File: new-test-recording
ActionID: 555544443333

Response: Success
ActionID: 555544443333
Message: Changed monitor filename
```

## Command

***Выполняет CLI-команду Asterisk***

Выполняет CLI-команду Asterisk так, как если бы она выполнялась из CLI.

### Параметры

`Command`

[обязательный] CLI-команда Asterisk, которая должна быть выполнена.

`ActionID`

[необязательный] Идентификатор команды, который может использоваться для опознавания ответа Asterisk.

### Привилегии

`command`, `all`

### Пример

```
Action: Command
Command: core show version
ActionID: 0123456789abcdef

Response: Follows
Privilege: Command
ActionID: 0123456789abcdef
Asterisk SVN-branch-1.4-r55869 built by jsmith @ hockey on a ppc running Linux
on 2007-02-21 16:55:26 UTC
--END COMMAND--
```

## DBGet

***Возвращает запись AstDB***

Эта команда извлекает значение из базы данных AstDB.

### Параметры

`Family`

[обязательный] Семейство ключей AstDB, из которого извлекается значение.

`Key`

[обязательный] Имя ключа AstDB.

`ActionID`

[необязательный] Идентификатор, который может использоваться для опознавания ответа на эту команду.

### Привилегии

system, all

### Пример

```
Action: DBGet
Family: testfamily
Key: mykey
ActionID: 01234-astdb-43210

Response: Success
Message: Result will follow
ActionID: 01234-astdb-43210

Event: DBGetResponse
Family: testfamily
Key: mykey
Val: 42
ActionID: 01234-astdb-43210
```

## DBPut

***Сохраняет запись в базе данных***

Задает значение ключа в базе данных AstDB.

### Параметры

Family

[обязательный] Семейство ключей AstDB, для которого задается значение.

Key

[обязательный] Имя ключа AstDB.

Val

[обязательный] Значение, которое должно быть присвоено ключу.

ActionID

[необязательный] Идентификатор, который может использоваться для опознавания ответа на эту команду.

### Привилегии

system, all

### Пример

```
Action: DBPut
Family: testfamily
Key: mykey
Val: 42
ActionID: testing123
```

```
Response: Success
Message: Updated database successfully
ActionID: testing123
```

## Events

***Управляет потоком событий***

Активирует или отключает отправку событий на это соединение интерфейса Manager.

### Параметры

*EventMask*

[обязательный] Задайте значение `on`, если должны отправляться все события, `off`, если события не должны передаваться, или `system`, `call`, `log`, чтобы выбрать тип событий, который должен отправляться на это соединение интерфейса Manager.

`ActionID`

[необязательный] Идентификатор, который может использоваться для опознавания ответа на эту команду.

### Привилегии

`none`

### Пример

```
Action: Events
EventMask: off
ActionID: 2938416

Response: Events Off
ActionID: 2938416

Action: Events
EventMask: log,call
ActionID: blah1234

Response: Events On
ActionID: blah1234
```

## ExtensionState

***Проверяет состояние добавочного номера***

Эта команда сообщает о состоянии заданного добавочного номера. Если добавочный номер имеет подсказку, эта команда обеспечит передачу состояния устройства, соединенного с данным добавочным номером.

## Параметры

`Exten`

[обязательный] Имя проверяемого добавочного номера.

`Context`

[обязательный] Имя контекста, в котором находится заданный добавочный номер.

`ActionId`

[необязательный] Идентификатор команды, который может использоваться для опознавания этой транзакции интерфейса Manager.

## Привилегии

`call`, `all`

## Пример

```
Action: ExtensionState
Exten: 200
Context: lab
ActionID: 54321

Response: Success
ActionID: 54321
Message: Extension Status
Exten: 200
Context: lab
Hint: SIP/testphone
Status: 0
```

## Примечания

Вот возможные состояния добавочного номера:

`-2`

Добавочный номер удален.

`-1`

Подсказка добавочного номера не обнаружена.

`0`

Свободен.

`1`

Используется.

`2`

Занят.

## GetConfig

***Возвращает конфигурацию***

Извлекает данные из конфигурационного файла Asterisk.

### Параметры

`Filename`

[обязательный] Имя конфигурационного файла, из которого должны извлекаться данные.

`ActionID`

[необязательный] Идентификатор, который может использоваться для опознавания ответа на эту команду.

### Привилегии

`config`, `all`

### Пример

```
Action: GetConfig
Filename: musiconhold.conf
ActionID: 09235012

Response: Success
ActionID: 09235012
Category-000000: default
Line-000000-000000: mode=files
Line-000000-000001: directory=/var/lib/asterisk/moh
Line-000000-000002: random=yes
```

## GetVar

***Возвращает значение переменной***

Возвращает значение локальной переменной канала или глобальной переменной.

### Параметры

`Channel`

[необязательный] Имя канала, значение переменной которого должно быть возвращено.

`Variable`

[обязательный] Имя переменной.

`ActionID`

[необязательный] Идентификатор, который может использоваться для опознавания ответа на эту команду.

### Привилегии

call, all

### Пример

```
Action: GetVar
Channel: SIP/linksys2-1020e2b0
Variable: SIPUSERAGENT
ActionID: abcd1234

Response: Success
Variable: SIPUSERAGENT
Value: Linksys/SPA962-5.1.5
ActionID: abcd1234

Action: GetVar
Variable: TRUNKMSD

Response: Success
Variable: TRUNKMSD
Value: 1
```

## Hangup

***Выполняет разъединение канала***

Выполняет разрыв соединения по заданному каналу.

### Параметры

Channel

[необязательный] Имя канала, разъединение которого должно быть выполнено.

ActionID

[необязательный] Идентификатор, который может использоваться для опознавания ответа на эту команду.

### Привилегии

call, all

### Пример

```
Action: Hangup
Channel: SIP/labrat-8d3a

Response: Success
Message: Channel Hungup

Event: Hangup
Privilege: call,all
```

```
Channel: SIP/labrat-8d3a
Uniqueid: 1173448206.0
Cause: 0
Cause-txt: Unknown
```

## IAXNetstats

***Представляет статистику IAX***

Представляет сводку статистки сетевой активности драйвера IAX2-канала.

### Привилегии

none

### Пример

```
Action: IAXNetstats

IAX2/216.207.245.8:4569-1 608 -1 0 -1 -1 0 -1 1 288 508 10 1 3 0 0
```

## IAXPeers

***Создает список равноправных участников IAX***

Создает список всех равноправных участников IAX2 с указанием их текущего статуса.

### Привилегии

none

### Пример

```
Action: IAXPeers

Name/Username   Host              Mask              Port    Status
jared/jared     192.168.0.71   (S) 255.255.255.255  4569    UNREACHABLE
jaredsmith      192.168.0.72   (S) 255.255.255.255  4569    OK (43 ms)
arrivaltel/8017 172.20.95.2    (S) 255.255.255.255  4569    Unmonitored
sokol/jsmith    172.17.122.217 (S) 255.255.255.255  4569    OK (48 ms)
demo/asterisk   216.207.245.47 (S) 255.255.255.255  4569    Unmonitored
5 iax2 peers [2 online, 1 offline, 2 unmonitored]
```

## ListCommands

***Создает список команд интерфейса Manager***

Создает список всех команд интерфейса Asterisk Manager с указанием имени команды и краткого описания каждой из них.

### Привилегии

none

### Пример

```
Action: ListCommands

Response: Success
AbsoluteTimeout: Set Absolute Timeout (Priv: call,all)
AgentCallbackLogin: Sets an agent as logged in by callback (Priv: agent,all)
AgentLogoff: Sets an agent as no longer logged in (Priv: agent,all)
. . .
ZapTransfer: Transfer Zap Channel (Priv: <none>)
```

## Logoff

***Завершает сеанс интерфейса Manager***

Завершает данный сеанс интерфейса Manager.

### Привилегии

none

### Пример

```
Action: Logoff

Response: Goodbye
Message: Thanks for all the fish.
```

## MailboxCount

***Проверяет количество сообщений в почтовом ящике***

Возвращает количество сообщений в заданном ящике голосовой почты.

### Привилегии

call, all

### Пример

```
Action: MailboxCount
Mailbox: 100@lab
ActionID: 54321abcde

Response: Success
ActionID: 54321abcde
Message: Mailbox Message Count
Mailbox: 100@lab
NewMessages: 2
OldMessages: 0
```

## MailboxStatus

***Проверяет статус почтового ящика***

Проверяет статус заданного ящика голосовой почты.

### Параметры

`Mailbox`

[обязательный] Полный идентификатор почтового ящика, включая почтовый ящик и контекст (*`ящик контекст`*).

`ActionID`

[необязательный] Уникальный идентификатор, который может использоваться для опознавания ответов на эту команду интерфейса Manager.

### Привилегии

`call`, `all`

### Пример

```
Action: MailboxStatus
Mailbox: 100@lab
ActionID: abcdef0123456789

Response: Success
ActionID: abcdef0123456789
Message: Mailbox Status
Mailbox: 100@lab
Waiting: 1
```

## MeetmeMute

***Выключает микрофон пользователя MeetMe***

Выключает микрофон конкретного пользователя, участвующего в конференции MeetMe.

### Параметры

`Meetme`

[обязательный] Номер конференции MeetMe.

`Usernum`

[обязательный] Номер абонента в заданной конференции.

`ActionID`

[необязательный] Уникальный идентификатор, который может служить для опознавания ответов на эту команду.

### Привилегии

`call, all`

### Пример

```
Action: MeetmeMute
Meetme: 104
Usernum: 1
ActionID: 5432154321

Response: Success
ActionID: 5432154321
Message: User muted

Event: MeetmeMute
Privilege: call,all
Channel: SIP/linksys2-10211dc0
Uniqueid: 1174008176.3
Meetme: 104
Usernum: 1
Status: on
```

### Примечания

Чтобы найти номер `Usernum` конкретного вызывающего абонента, понаблюдайте за интерфейсом Asterisk Manager при входе нового участника в конференцию. Когда это произойдет, вы увидите такое событие:

```
Event: MeetmeJoin
Privilege: call,all
Channel: SIP/linksys2-10211dc0
Uniqueid: 1174008176.3
Meetme: 104
Usernum: 1
```

## MeetMeUnmute

***Включает микрофон пользователя MeetMe***

Включает микрофон заданного пользователя в конференции MeetMe.

### Параметры

`Meetme`

**[обязательный]** Номер конференции MeetMe.

`Usernum`

**[обязательный]** Номер пользователя в заданной конференции.

`ActionID`

**[необязательный]** Уникальный идентификатор, который может служить для опознавания ответов на эту команду.

### Привилегии

call, all

### Пример

```
Action: MeetmeUnmute
Meetme: 104
Usernum: 1
ActionID: abcdefghijklmnop

Response: Success
ActionID: abcdefghijklmnop
Message: User unmuted

Event: MeetmeMute
Privilege: call,all
Channel: SIP/linksys2-10211dc0
Uniqueid: 1174008176.3
Meetme: 104
Usernum: 1
Status: off
```

## Monitor

***Записывает разговор по каналу***

Записывает аудиоданные, передаваемые по каналу, в заданный файл.

### Параметры

Channel

[обязательный] Определяет канал, разговор по которому будет записываться.

File

[необязательный] Имя файла, в который производится запись разговора по каналу. По умолчанию запись ведется в папку очереди для записи разговоров Asterisk, которой обычно является /var/spool/asterisk/monitor. Если имя файла не задано, в качестве него используется имя канала, при этом символы слэша заменяются на тире.

Format

[необязательный] Аудиоформат, в котором производится запись канала. По умолчанию используется wav.

Mix

[необязательный] Логический флаг, определяющий, должна ли Asterisk объединять входящий и исходящий аудиопотоки канала в один файл.

ActionID

[необязательный] Идентификатор, который может использоваться для опознавания ответа на эту команду.

### Привилегии

call, all

### Пример

```
Action: Monitor
Channel: SIP/linksys2-10216e38
Filename: test-recording
Format: gsm
Mix: true

Response: Success
Message: Started monitoring channel
```

## Originate

***Формирует вызов***

Формирует исходящий вызов из Asterisk и соединяет канал с контекстом/добавочным номером/приоритетом или приложением диалплана.

### Параметры

Channel

[обязательный] Имя канала, которому адресован вызов. Как только вызываемый канал отвечает на вызов, управление вызовом передается в заданные Exten/Context/Priority или Application.

Exten

[необязательный] Используемый добавочный номер (должны быть заданы значения Context и Priority).

Context

[необязательный] Используемый контекст (должны быть заданы значения Exten и Priority).

Priority

[необязательный] Используемый приоритет (должны быть заданы значения Exten и Context).

Application

[необязательный] Используемое приложение.

Data

[необязательный] Данные, которые должны быть переданы как параметры приложения (должно быть задано значение Application).

Timeout

[необязательный] Как долго необходимо ожидать ответа на звонок, в миллисекундах.

CallerID

[необязательный] Идентификатор вызывающего абонента, который должен быть задан для исходящего канала.

Variable

[необязательный] Переменная канала, которая должна быть задана. Допускается множество переменных в заголовке.

Account

[необязательный] Учетная запись.

Async

[необязательный] Задайте значение true, чтобы выполнять асинхронные вызовы. Асинхронное формирование вызовов позволяет создавать один или более вызовов, не ожидая немедленного ответа.

ActionID

[необязательный] Идентификатор, который может использоваться для опознавания ответа на эту команду.

## Привилегии

call, all

## Пример

```
Action: Originate
Channel: SIP/linksys2
Context: lab
Exten: 201
Priority: 1
CallerID:

Response: Success
Message: Originate successfully queued

Action: Originate
Application: MusicOnHold
Data: default
Channel: SIP/linksys2

Response: Success
Message: Originate successfully queued
```

# Park

***Выполняет парковку канала***

Выполняет парковку заданного канала в слот парковки.

## Параметры

`Channel`

[обязательный] Имя канала, который должен быть припаркован.

`Channel2`

[обязательный] Канал, которому должна быть предоставлена информация о парковке (и возвращен звонок, если допустимое время парковки истекло).

`Timeout`

[необязательный] Время ожидания, в миллисекундах, перед выполнением обратного вызова.

`ActionID`

[необязательный] Идентификатор, который может использоваться для опознавания ответа на эту команду.

## Привилегии

`call`, `all`

## Пример

```
Action: Park
Channel: SIP/linksys-10228fb0
Channel2: SIP/linksys2-10231520
Timeout: 45
ActionID: parking-test-01

Response: Success
ActionID: parking-test-01
Message: Park successful
```

## Примечания

Конфигурация слота парковки вызовов выполняется в файле features.conf в папке конфигурации Asterisk.

# ParkedCalls

***Создает список припаркованных вызовов***

Создает список всех вызовов, припаркованных в слоте парковки вызовов.

### Привилегии

none

### Пример

```
Action: ParkedCalls
ActionID: 0982350175

Response: Success
ActionID: 0982350175
Message: Parked calls will follow

Event: ParkedCall
Exten: 701
Channel: SIP/linksys2-101f98a8
From: SIP/linksys2-101f98a8
Timeout: 26
CallerID: linksys2
CallerIDName: linksys2
ActionID: 0982350175

Event: ParkedCallsComplete
ActionID: 0982350175
```

### Примечания

Конфигурация слота парковки вызовов выполняется в файле features.conf в папке конфигурации Asterisk.

## PauseMonitor

***Приостанавливает запись канала***

Приостанавливает отслеживание (запись) канала, для которого производится запись разговоров.

### Параметры

Channel

[обязательный] Идентификатор записываемого в настоящее время канала.

ActionID

[необязательный] Идентификатор, который может использоваться для опознавания ответа на эту команду.

### Привилегии

call, all

### Пример

```
Action: PauseMonitor
Channel: SIP/linksys2-10212040
ActionID: 987987987987

Response: Success
ActionID: 987987987987
Message: Paused monitoring of the channel
```

## Ping

***Поддерживает соединение активным***

Посылает запрос на сервер Asterisk, чтобы убедиться, что он до сих пор отвечает. Asterisk ответит сообщением `Pong`. Эта команда также может использоваться, чтобы не допустить разрыва соединения в результате истечения времени ожидания.

### Пример

```
Action: Ping

Response: Pong
```

## PlayDTMF

***Передает DTMF-код в канал***

Передает DTMF-код в заданный канал.

### Параметры

`Channel`

[обязательный] Идентификатор канала, в который должен быть отправлен DTMF-код.

`Digit`

[обязательный] DTMF-код, который должен быть отправлен в канал.

`ActionID`

[необязательный] Идентификатор, который может использоваться для опознавания ответа на эту команду.

### Привилегии

`call`, `all`

### Пример

```
Action: PlayDTMF
Channel: Local/201@lab-157a,1
Digit: 9
```

```
Response: Success
Message: DTMF successfully queued
```

## QueueAdd

***Добавляет участника в заданную очередь***

Добавляет участника обработки очереди вызовов.

### Параметры

`Queue`

[обязательный] Имя очереди вызовов.

`Interface`

[обязательный] Имя участника, который должен быть добавлен в очередь. Это будет технология или ресурс, например `SIP/Jane` или `Local/203@lab/n`. Также с помощью синтаксиса `Agent/1234` могут быть добавлены агенты (описанные в файле agents.conf).

`MemberName`

[необязательный] Это удобный для человека псевдоним интерфейса, который будет использоваться в статистических данных и журналах регистрации очереди.

`Penalty`

[необязательный] Числовой приоритет, применяемый к данному участнику обработки очереди. Asterisk сначала пытается распределить вызовы между участниками с более низкими значениями приоритетов, а затем переходит к участникам с более высокими приоритетами.

`Paused`

[необязательный] Должен ли участник добавляться в изначально приостановленном состоянии.

`ActionID`

[необязательный] Идентификатор команды, который можно использовать для опознавания ответа на эту транзакцию интерфейса Manager.

### Привилегии

`agent`, `all`

### Пример

```
Action: QueueAdd
Queue: myqueue
Interface: SIP/testphone
```

```
MemberName: Jared Smith
Penalty: 2
Paused: no
ActionID: 4242424242

Response: Success
ActionID: 4242424242
Message: Added interface to queue

Event: QueueMemberAdded
Privilege: agent,all
Queue: myqueue
Location: SIP/testphone
MemberName: Jared Smith
Membership: dynamic
Penalty: 2
CallsTaken: 0
LastCall: 0
Status: 1
Paused: 0
```

## QueuePause

***Приостанавливает или возобновляет работу участника обработки очереди вызовов***

Приостанавливает или возобновляет работу участника обработки очереди вызовов.

### Параметры

`Interface`

[обязательный] Имя интерфейса, участие которого в обработке вызовов должно быть приостановлено или возобновлено.

`Paused`

[обязательный] Должна ли быть приостановлена работа участника обработки вызовов. Задайте значение `true`, чтобы приостановить работу участника, или `false`, чтобы возобновить ее.

`Queue`

[необязательный] Имя очереди вызовов, участие в обработке которой приостанавливается или возобновляется для данного участника. Если не задано, работа участника будет приостановлена или возобновлена во всех очередях вызовов, участником обработки которых он является.

`ActionID`

[необязательный] Идентификатор, который может использоваться для опознавания ответа на эту команду.

### Привилегии

agent, all

### Пример

```
Action: QueuePause
Interface: SIP/testphone
Paused: true
Queue: myqueue

Response: Success
Message: Interface paused successfully

Event: QueueMemberPaused
Privilege: agent,all
Queue: myqueue
Location: SIP/testphone
MemberName: Jared Smith
Paused: 1

Action: QueuePause
Interface: SIP/testphone
Paused: false

Response: Success
Message: Interface unpaused successfully

Event: QueueMemberPaused
Privilege: agent,all
Queue: myqueue
Location: SIP/testphone
MemberName: Jared Smith
Paused: 0
```

## QueueRemove

***Удаляет интерфейс из очереди***

Удаляет интерфейс из очереди вызовов.

### Параметры

Queue

[обязательный] Очередь, из которой должен быть удален участник.

Interface

[обязательный] Интерфейс (участник), который должен быть удален из заданной очереди.

ActionID

[необязательный] Идентификатор, который может использоваться для опознавания ответа на эту команду.

### Привилегии

agent, all

### Пример

```
Action: QueueRemove
Queue: myqueue
Interface: SIP/testphone

Response: Success
Message: Removed interface from queue

Event: QueueMemberRemoved
(Privilege: agent,all
Queue: myqueue
Location: SIP/testphone
MemberName: Jared Smith
```

## QueueStatus

***Проверяет статус очереди***

Проверяет статус одной или более очередей вызовов.

### Параметры

Queue

[необязательный] Если задан, ограничивает ответ статусом заданной очереди.

Member

[необязательный] Идентификатор команды, который можно использовать для опознавания ответа на эту транзакцию интерфейса Manager.

ActionID

[необязательный] Идентификатор, который может использоваться для опознавания ответа на эту команду.

### Привилегии

none

### Пример

```
Action: QueueStatus
Queue: inbound-queue
ActionID: 11223344556677889900

Response: Success
ActionID: 11223344556677889900
Message: Queue status will follow
```

```
Event: QueueParams
Queue: inbound-queue
Max: 0
Calls: 1
Holdtime: 99
Completed: 540
Abandoned: 51
ServiceLevel: 60
ServicelevelPerf: 50.4
Weight: 0
ActionID: 11223344556677889900

Event: QueueMember
Queue: inbound-queue
Location: Local/4020@agents/n
Membership: dynamic
Penalty: 2
CallsTaken: 25
LastCall: 1175563440
Status: 2
Paused: 0
ActionID: 11223344556677889900

Event: QueueEntry
Queue: inbound-queue
Position: 1
Channel: Zap/25-1
CallerID: 8012317154
CallerIDName: JOHN Q PUBLIC
Wait: 377
ActionID: 11223344556677889900

Event: QueueStatusComplete
ActionID: 11223344556677889900
```

## Queues

***Представляет основную информацию очередей вызовов***

**Представляет очереди вызовов с информацией об участниках обработки очередей, вызывающих абонентах и с основными статистическими данными.**

### Привилегии

none

### Пример

```
Action: Queues

inbound-queue has 0 calls (max unlimited) in 'rrmemory' strategy (81s holdtime),
  W:0, C:542, A:51, SL:50.4% within 60s
```

```
Members:
   Local/4020@agents/n with penalty 2 (dynamic) (Unknown) has taken
      27 calls (last was 124 secs ago)
No Callers
```

### Примечания

Эта команда интерфейса Manager обеспечивает вывод, аналогичный выводу команды `show queues` интерфейса командной строки Asterisk. Однако вывод данной команды трудно поддается программному синтаксическому разбору, поэтому, вероятно, лучше использовать команду `QueueStatus`.

## Redirect

***Перенаправляет (переадресовывает) канал***

Перенаправляет канал в новый контекст, добавочный номер и приоритет диалплана.

### Параметры

`Channel`

[обязательный] Перенаправляемый канал.

`ExtraChannel`

[необязательный] Идентификатор канала второго плеча вызова для переадресации.

`ActionID`

[необязательный] Идентификатор, который может использоваться для опознавания ответа на эту команду.

`Exten`

[обязательный] Добавочный номер в диалплане, на который выполняется переадресация.

`Context`

[обязательный] Контекст, на который выполняется переадресация.

`Priority`

[обязательный] Приоритет, на который выполняется переадресация.

### Привилегии

`call`, `all`

### Пример

```
Action: Redirect
Channel: SIP/linksys2-10201e90
Context: lab
```

```
Exten: 500
Priority: 1
ActionID: 010123234545

Response: Success
ActionID: 010123234545
Message: Redirect successful
```

## SIPpeers

***Создает список всех равноправных участников SIP***

Создает список сконфигурированных в данный момент равноправных участников SIP с указанием их статуса.

### Параметры

ActionID

[необязательный] Идентификатор команды, который можно использовать для опознавания ответа на эту транзакцию интерфейса Manager.

### Привилегии

system, all

### Пример

```
Action: SIPPeers
ActionID: 555444333222111

Response: Success
ActionID: 555444333222111
Message: Peer status list will follow

Event: PeerEntry
ActionID: 555444333222111
Channeltype: SIP
ObjectName: labrat
ChanObjectType: peer
IPaddress: 10.0.0.75
IPport: 5060
Dynamic: no
Natsupport: no
VideoSupport: no
ACL: no
Status: OK (318 ms)
RealtimeDevice: no

Event: PeerEntry
ActionID: 555444333222111
Channeltype: SIP
```

```
ObjectName: guineapig
ChanObjectType: peer
IPaddress: 172.18.227.72
IPport: 5060
Dynamic: no
Natsupport: no
VideoSupport: no
ACL: no
Status: Unmonitored
RealtimeDevice: no

Event: PeerEntry
ActionID: 555444333222111
Channeltype: SIP
ObjectName: another
ChanObjectType: peer
IPaddress: 172.18.227.73
IPport: 5060
Dynamic: yes
Natsupport: no
VideoSupport: no
ACL: no
Status: Unmonitored
RealtimeDevice: no

Event: PeerlistComplete
ListItems: 7
ActionID: 555444333222111
```

## SIPShowPeer

***Представляет информацию о равноправном участнике SIP***

Представляет подробную информацию о сконфигурированном равноправном участнике SIP.

### Параметры

`Peer`

[обязательный] Имя равноправного участника SIP.

`ActionID`

[необязательный] Идентификатор команды, который можно использовать для опознавания ответа на эту транзакцию интерфейса Manager.

### Привилегии

`system`, `all`

### Пример

```
Action: SIPShowPeer
Peer: linksys2
ActionID: 9988776655

Response: Success
ActionID: 9988776655
Channeltype: SIP
ObjectName: linksys2
ChanObjectType: peer
SecretExist: Y
MD5SecretExist: N
Context: lab
Language:
AMAflags: Unknown
CID-CallingPres: Presentation Allowed, Not Screened
Callgroup:
Pickupgroup:
VoiceMailbox:
TransferMode: open
LastMsgsSent: -1
Call-limit: 0
MaxCallBR: 384 kbps
Dynamic: Y
Callerid: "Linksys #2" <555>
RegExpire: 2516 seconds
SIP-AuthInsecure: no
SIP-NatSupport: RFC3581
ACL: N
SIP-CanReinvite: Y
SIP-PromiscRedir: N
SIP-UserPhone: N
SIP-VideoSupport: N
SIP-DTMFmode: rfc2833
SIPLastMsg: 0
ToHost:
Address-IP: 192.168.5.71
Address-Port: 5061
Default-addr-IP: 0.0.0.0
Default-addr-port: 5056
Default-Username: linksys2
RegExtension: 6100
Codecs: 0x4 (ulaw)
CodecOrder: ulaw
Status: Unmonitored
SIP-Useragent: Linksys/SPA962-5.1.5
Reg-Contact : sip:linksys2@192.168.5.71:5061
```

## SetCDRUserField

***Задает поле пользователя записи CDR***

Задает настройку `UserField` записи CDR для указанного канала.

### Параметры

Channel

[обязательный] Канал, для которого задается настройка UserField записи CDR.

UserField

[обязательный] Значение, которое должно быть присвоено UserField в записи CDR.

ActionID

[необязательный] Идентификатор, который может использоваться для опознавания ответа на эту команду.

### Привилегии

call, all

### Пример

```
Action: SetCDRUserField
Channel: SIP/test-10225140
UserField: abcdefg

Response: Success
Message: CDR Userfield Set
```

## SetVar

***Задает переменную канала***

Задает значение глобальной переменной или переменной канала.

### Параметры

Channel

[необязательный] Канал, для переменной которого задается значение. Если не указан, переменная будет задана как глобальная.

Variable

[обязательный] Имя переменной.

Value

[обязательный] Значение.

ActionID

[необязательный] Идентификатор, который может использоваться для опознавания ответа на эту команду.

### Привилегии

call, all

### Пример

```
Action: SetVar
Channel: SIP/linksys2-10225140
Variable: MyOwnChannelVariable
Value: 42

Response: Success
Message: Variable Set

Action: SetVar
Variable: MyOwnGlobalVariable
Value: 25

Response: Success
Message: Variable Set
```

## Status

***Представляет статус канала***

Представляет статус одного или более каналов с подробной информацией об их текущем состоянии.

### Параметры

`Channel`

[необязательный] Ограничивает вывод статусом заданного канала.

`ActionID`

[необязательный] Идентификатор команды, который можно использовать для опознаввания ответа на эту транзакцию интерфейса Manager.

### Привилегии

`call`, `all`

### Пример

```
Action: Status
Channel: SIP/test-10225140
ActionID: 101010101010101

Response: Success
ActionID: 101010101010101
Message: Channel status will follow

Event: Status
Privilege: Call
Channel: SIP/test-10225140
CallerID: "Bob Jones" <501>
CallerIDNum: 501
CallerIDName: "Bob Jones"
```

```
Account:
State: Up
Context: lab
Extension: 201
Priority: 1
Seconds: 865
Link: Local/200@lab-4d13,1
Uniqueid: 1177550165.0
ActionID: 101010101010101
Event: StatusComplete
ActionID: 101010101010101
```

## StopMonitor

***Прекращает запись разговора по каналу***

Прекращает ранее начатое отслеживание (запись) канала.

### Параметры

`Channel`

[обязательный] Имя канала, запись которого должна быть прекращена.

`ActionID`

[необязательный] Уникальный идентификатор, который может служить для опознавания ответов на эту команду.

### Привилегии

`call`, `all`

### Пример

```
Action: StopMonitor
Channel: SIP/linksys2-10216e38

Response: Success
Message: Stopped monitoring channel
```

## UnpauseMonitor

***Возобновляет запись***

Возобновляет отслеживание (запись) разговора по указанному каналу.

### Параметры

`Channel`

[обязательный] Имя канала, запись которого должна быть возобновлена.

`ActionID`

[необязательный] Уникальный идентификатор, который может служить для опознавания ответов на эту команду.

### Привилегии

`call, all`

### Пример

```
Action: UnpauseMonitor
Channel: SIP/linksys2-10212040
ActionID: 282828282828282

Response: Success
ActionID: 282828282828282
Message: Unpaused monitoring of the channel
```

## UpdateConfig

***Обновляет конфигурационный файл***

Динамически обновляет конфигурационный файл Asterisk.

### Параметры

`SrcFilename`

[обязательный] Имя конфигурационного файла, из которого следует читать текущую информацию.

`DstFilename`

[обязательный] Имя записываемого конфигурационного файла.

`Reload`

[необязательный] Определяет, должна ли быть выполнена перезагрузка после обновления конфигурации, или задает имя конкретного модуля, который должен быть перезагружен.

`Action-`*`XXXXXX`*

[обязательный] Действие, которое необходимо предпринять. Это может быть `NewCat`, `RenameCat`, `DelCat`, `Update`, `Delete` или `Append`.

`Cat-`*`XXXXXX`*

[обязательный] Имя изменяемой категории.

`Var-`*`XXXXXX`*

[необязательный] Имя изменяемой переменной.

`Value-`*`XXXXXX`*

[необязательный] Значение изменяемой переменной.

`Match-`*`XXXXXX`*

[необязательный] Если задан, является дополнительным параметром, которому должен соответствовать параметр линии.

`ActionID`

[необязательный] Идентификатор, который может использоваться для опознавания ответа на эту команду.

### Привилегии

`config`, `all`

### Пример

```
Action: UpdateConfig
SrcFilename: sip.conf
DstFilename: test.conf
Action-000000: update
Cat-000000: linksys
Var-000000: mailbox
Value-000000: 101@lab

Response: Success
```

### Примечания

Обратите внимание, что первый набор параметров должен быть обозначен номером `000000`, второй – `000001` и т. д. Это позволяет одновременно обновлять множество разных значений конфигурации. Также следует отметить, что Asterisk GUI использует это как основной механизм для обновления конфигурации Asterisk.

## UserEvent

***Отправляет произвольное событие***

Отправляет произвольное событие в интерфейс Asterisk Manager.

### Параметры

`UserEvent`

[обязательный] Имя отправляемого произвольного события.

`Header`

[необязательный] Имя и значение произвольного параметра вашего события. В событие можно ввести неограниченное число дополнительных заголовков (и их значений).

`ActionID`

[необязательный] Идентификатор, который может использоваться для опознавания ответа на эту команду.

### Привилегии

`user`, `all`

### Пример

```
Action: UserEvent
Blah: one
SomethingElse: two
ActionID: 63346

Event: UserEvent
Privilege: user,all
UserEvent:
Action: UserEvent
Blah: one
SomethingElse: two
ActionID: 63346
```

## WaitEvent

***Ожидает возникновения события***

После вызова этой команды Asterisk отправит сообщение `Success`, как только интерфейс Asterisk Manager поставит в очередь следующее событие. Если команда `WaitEvent` вызвана для HTTP-сеанса интерфейса Manager, события будут формироваться и ставиться в очередь.

### Параметры

`Timeout`

[необязательный] Максимальное время ожидания событий.

`ActionID`

[необязательный] Идентификатор, который может использоваться для опознавания ответа на эту команду.

### Привилегии

`none`

### Пример

```
Action: WaitEvent
Timeout: 30

Action: Ping

Response: Success
Message: Waiting for Event...
Event: WaitEventComplete

Response: Pong
```

## ZapDNDoff

***Отменяет для Zap-канала состояние «не беспокоить»***

Отменяет для Zap-канала состояние «не беспокоить».

### Параметры

ZapChannel

[обязательный] Номер Zap-канала, для которого необходимо отменить состояние «не беспокоить».

ActionID

[необязательный] Идентификатор, который может использоваться для опознавания ответа на эту команду.

### Привилегии

none

### Пример

```
Action: ZapDNDoff
ZapChannel: 1
ActionID: 01234567899876543210

Response: Success
ActionID: 01234567899876543210
Message: DND Disabled
```

## ZapDNDon

***Устанавливает для Zap-канала состояние «не беспокоить»***

Включает для заданного Zap-канала состояние «не беспокоить».

### Параметры

ZapChannel

[обязательный] Номер Zap-канала, который необходимо перевести в состояние «не беспокоить».

ActionID

[необязательный] Идентификатор, который может использоваться для опознавания ответа на эту команду.

### Привилегии

none

### Пример

```
Action: ZapDNDon
ZapChannel: 1
ActionID: 98765432100123456789

Response: Success
ActionID: 98765432100123456789
Message: DND Enabled
```

## ZapDialOffhook

***Выполняет набор номера по Zap-каналу, когда телефон подключен к линии***

Набирает заданный номер по Zap-каналу, когда телефон подключен к линии.

### Параметры

`ZapChannel`

[обязательный] Zap-канал, по которому выполняется набор номера.

`Number`

[обязательный] Набираемый номер.

`ActionID`

[необязательный] Уникальный идентификатор, который может служить для опознавания ответов на эту команду.

### Привилегии

`none`

### Пример

```
Action: ZapDialOffhook
ZapChannel: 1
Number: 543215432154321
ActionID: 5676

Response: Success
ActionID: 5676
Message: ZapDialOffhook
```

## ZapHangup

***Разъединяет Zap-канал***

Выполняет разъединение связи по заданному Zap-каналу.

### Параметры

*ZapChannel*

[обязательный] Zap-канал, который должен быть разъединен.

*ActionID*

[необязательный] Уникальный идентификатор, который может служить для опознавания ответов на эту команду.

### Привилегии

none

### Пример

```
Action: ZapHangup
ZapChannel: 1-1
ActionID: 98237892

Response: Success
ActionID: 98237892
Message: ZapHangup
```

## ZapRestart

***Выполняет полный перезапуск Zaptel-каналов***

Полностью перезапускает Zaptel-каналы, завершая все выполняющиеся вызовы.

### Привилегии

none

### Пример

```
Action: ZapRestart

Response: Success
Message: ZapRestart: Success
```

## ZapShowChannels

***Представляет статус Zapata-каналов***

Представляет статус всех Zap-каналов.

### Параметры

*ActionID*

[необязательный] Идентификатор команды, который может использоваться для опознавания ответа Asterisk.

### Привилегии

none

### Пример

```
Action: ZapShowChannels
ActionID: 9999999999

Response: Success
ActionID: 9999999999
Message: Zapata channel status will follow

Event: ZapShowChannels
Channel: 1
Signalling: FXO Kewlstart
Context: incoming
DND: Disabled
Alarm: No Alarm
ActionID: 9999999999

Event: ZapShowChannels
Channel: 4
Signalling: FXS Kewlstart
Context: incoming
DND: Disabled
Alarm: No Alarm
ActionID: 9999999999

Event: ZapShowChannelsComplete
ActionID: 9999999999
```

## ZapTransfer

***Выполняет переадресацию Zap-канала***

Выполняет переадресацию Zap-канала.

### Привилегия

none

### Пример

```
Action: ZapTransfer
ZapChannel: 1
ActionID: 4242

Response: Success
Message: ZapTransfer
ActionID: 4242
```

# G

# Пример func_odbc

В этом приложении представлены примеры из реализации функциональности «горячих столов», о которой шла речь в разделе «Ощутим могущество func_odbc: система “горячих столов”» главы 12. Возможно, есть смысл вернуться к главе 12 и рассмотреть этот код данного примера вместе с объяснениями.

## Система «горячих столов» (extensions.conf)

***Код диалплана***

```
; Функция "горячих столов"
[hotdesk]
; Регистрация "горячего стола"
exten => _11XX,1,NoOp()
exten => _11XX,n,Set(E=${EXTEN})
exten => _11XX,n,Verbose(1|Hot Desk Extension ${E} is changing status)
exten => _11XX,n,Verbose(1|Checking current status of extension ${E})
exten => _11XX,n,Set(${E}_STATUS=${HOTDESK_INFO(status,${E})})
exten => _11XX,n,Set(${E}_PIN=${HOTDESK_INFO(pin,${E})})
exten => _11XX,n,GotoIf($[${ISNULL(${${E}_STATUS})}]?invalid_user)

exten => _11XX,n,GotoIf($[${${E}_STATUS} = 1]?logout,1:login,1)

exten => login,1,NoOp()
exten => login,n,Set(PIN_TRIES=0)
exten => login,n,Set(MAX_PIN_TRIES=3)
exten => login,n(get_pin),NoOp()
exten => login,n,Set(PIN_TRIES=$[${PIN_TRIES} + 1])
exten => login,n,Read(PIN_ENTERED|enter-password|${LEN(${${E}_PIN})})
```

```
exten => login,n,GotoIf($[${PIN_ENTERED} = ${${E}_PIN}]?valid_login,1)
exten => login,n,Playback(invalid-pin,1)
exten => login,n,GotoIf($[${PIN_TRIES} <= ${MAX_PIN_TRIES}]?get_pin:login_fail,1)

exten => valid_login,1,NoOp()
exten => valid_login,n,Set(LOCATION=${CUT(CHANNEL,/,2)})
exten => valid_login,n,Set(LOCATION=${CUT(LOCATION,-,1)})
exten => valid_login,n,Set(ARRAY(USERS_LOGGED_IN)=${HOTDESK_CHECK_PHONE_LOGINS
(${LOCATION})})
exten => valid_login,n,GotoIf($[${USERS_LOGGED_IN} > 0]?logout_login,1)
exten => valid_login,n(set_login_status),NoOp()
exten => valid_login,n,Set(HOTDESK_STATUS(${E})=1\,${LOCATION})

exten => valid_login,n,GotoIf($[${ODBCROWS} < 1]?error,1)
exten => valid_login,n,Playback(agent-loginok)
exten => valid_login,n,Hangup()

exten => logout_login,1,NoOp()
exten => logout_login,n,Set(ROW_COUNTER=0)
exten => logout_login,n,While($[${ROW_COUNTER} < ${USERS_LOGGED_IN}])
exten => logout_login,n,Set(WHO=${HOTDESK_LOGGED_IN_USER(${LOCATION},
${ROW_COUNTER})})
exten => logout_login,n,Set(HOTDESK_STATUS(${WHO})=0)
exten => logout_login,n,Set(ROW_COUNTER=$[${ROW_COUNTER} + 1])
exten => logout_login,n,EndWhile()
exten => logout_login,n,Goto(valid_login,set_login_status)

exten => logout,1,NoOp()
exten => logout,n,Set(HOTDESK_STATUS(${E})=0)
exten => logout,n,GotoIf($[${ODBCROWS} < 1]?error,1)
exten => logout,n,Playback(silence/1&agent-loggedoff)
exten => logout,n,Hangup()

exten => login_fail,1,NoOp()
exten => login_fail,n,Playback(silence/1&login-fail)
exten => login_fail,n,Hangup()

exten => error,1,NoOp()
exten => error,n,Playback(silence/1&connection-failed)
exten => error,n,Hangup()

exten => invalid_user,1,NoOp()
exten => invalid_user,n,Verbose(1|Hot Desk extension ${E} does not exist)
exten => invalid_user,n,Playback(silence/2&invalid)
exten => invalid_user,n,Hangup()

include => hotdesk_outbound

[hotdesk_outbound]
exten => _X.,1,NoOp()
exten => _X.,n,Set(LOCATION=${CUT(CHANNEL,/,2)})
```

```
exten => _X.,n,Set(LOCATION=${CUT(LOCATION,-,1)})
exten => _X.,n,Set(WHO=${HOTDESK_PHONE_STATUS(${LOCATION})})
exten => _X.,n,GotoIf($[${ISNULL(${WHO})}]?no_outgoing,1)
exten => _X.,n,Set(${WHO}_CID_NAME=${HOTDESK_INFO(cid_name,${WHO})})
exten => _X.,n,Set(${WHO}_CID_NUMBER=${HOTDESK_INFO(cid_number,${WHO})})
exten => _X.,n,Set(${WHO}_CONTEXT=${HOTDESK_INFO(context,${WHO})})
exten => _X.,n,Goto(${${WHO}_CONTEXT},${EXTEN},1)

[international]
exten => _011.,1,NoOp()
exten => _011.,n,Set(E=${EXTEN})
exten => _011.,n,Goto(outgoing,call,1)
exten => i,1,NoOp()
exten => i,n,Playback(silence/2&sorry-cant-let-you-do-that2)
exten => i,n,Hangup()

include => longdistance

[longdistance]
exten => _1NXXNXXXXXX,1,NoOp()
exten => _1NXXNXXXXXX,n,Set(E=${EXTEN})
exten => _1NXXNXXXXXX,n,Goto(outgoing,call,1)

exten => _NXXNXXXXXX,1,Goto(1${EXTEN},1)

exten => i,1,NoOp()
exten => i,n,Playback(silence/2&sorry-cant-let-you-do-that2)
exten => i,n,Hangup()

include => local

[local]
exten => _416NXXXXXX,1,NoOp()
exten => _416NXXXXXX,n,Set(E=${EXTEN})
exten => _416NXXXXXX,n,Goto(outgoing,call,1)

exten => i,1,NoOp()
exten => i,n,Playback(silence/2&sorry-cant-let-you-do-that2)
exten => i,n,Hangup()

[outgoing]
exten => call,1,NoOp()
exten => call,n,Set(CALLERID(name)=${${WHO}_CID_NAME})
exten => call,n,Set(CALLERID(number)=${${WHO}_CID_NUMBER})
exten => call,n,Dial(SIP/service_provider/${E})
exten => call,n,Playback(silence/2&pls-try-call-later)
exten => call,n,Hangup()

[hotdesk_phones]
exten => _11XX,1,NoOp()
exten => _11XX,n,Set(E=${EXTEN})
```

```
exten => _11XX,n,Set(LOCATION=${HOTDESK_LOCATION(${E})})
exten => _11XX,n,GotoIf($[${ISNULL(${LOCATION})}]?voicemail,1)
exten => _11XX,n,Dial(SIP/${LOCATION},30)
exten => _11XX,n,Goto(voicemail,1)

exten => voicemail,1,NoOp()
exten => voicemail,n,Voicemail(${E}@hotdesk,u)
exten => voicemail,n,Hangup()
```

### Смотрите также

Разделы «Система “горячих столов” (sip.conf)», «Система “горячих столов” (func_odbc.conf)», главу 5, главу 6, `Read()`, `CUT`, `While()`, `ISNULL`, `VoiceMail()`, `CALLERID`, `Dial()`, `GotoIf()`

## Система «горячих столов» (func_odbc.conf)

***Специальные функции диалплана***

```
[INFO]
prefix=HOTDESK
dsn=asterisk
read=SELECT ${ARG1} FROM ast_hotdesk WHERE extension = '${ARG2}'

[STATUS]
prefix=HOTDESK
dsn=asterisk
write=UPDATE ast_hotdesk SET status = '${VAL1}', location = '${VAL2}'
WHERE extension = '${ARG1}'

[CHECK_PHONE_LOGINS]
prefix=HOTDESK
dsn=asterisk
read=SELECT COUNT(status) FROM ast_hotdesk WHERE status = '1' AND location
= '${ARG1}'

[LOGGED_IN_USER]
prefix=HOTDESK
dsn=asterisk
read=SELECT extension FROM ast_hotdesk WHERE status = '1' AND location
= '${ARG1}'
ORDER BY id LIMIT '1' OFFSET '${ARG2}'

[PHONE_STATUS]
prefix=HOTDESK
dsn=asterisk
read=SELECT extension FROM ast_hotdesk WHERE location = '${ARG1}'
AND status = '1'
```

### Смотрите также

Разделы «Система “горячих столов” (extensions.conf)», «Система “горячих столов” (sip.conf)», res_odbc.conf

## Система «горячих столов» (sip.conf)

***Два примера конфигурации телефонов и пример конфигурации поставщика сервиса***

```
; ПОЛЬЗОВАТЕЛИ СИСТЕМЫ "ГОРЯЧИХ СТОЛОВ"
[desk_1]
type=friend
host=dynamic
secret=my_special_secret
context=hotdesk
qualify=yes

[desk_2]
type=friend
host=dynamic
secret=my_special_secret
context=hotdesk
qualify=yes

; КОНЕЦ ОПИСАНИЯ ПОЛЬЗОВАТЕЛЕЙ СИСТЕМЫ "ГОРЯЧИХ СТОЛОВ"
```

### Смотрите также

Разделы «Система “горячих столов” (extensions.conf)», «Система “горячих столов” (func_odbc.conf)», главу 4

# Алфавитный указатель

## Специальные символы



## A

## B



## C



## D

## E



## F

## G



## H



## I

## J



## K



## L

## M



## N

## O



## P



## Q

## R



## S

## T

## U



## V

## W



## X



## Y



## Z



## А

## Б



## В



## Г



## Д

## Е



## Ж



## З



## И



## К



## Л

## М



## Н



## О



## П

## P



## C

## Т



## У



## Ф



## Х



## Ц

## Ч



## Ш



## Э